

Master Claude Code: Proven Daily Workflows from 3 Technical Founders:

<https://www.youtube.com/watch?v=hOqgFNlbrYE>

- Best MCPs
- Context Forking
- Automated Code Review
- Top CodeGen Tools
- Prompting Tricks
- Validation Steps
- Others

"Your CLAUDE.md files become part of Claude's prompts, so they should be refined like any frequently used prompt" - Claude Code Documentation

Setup Tips

"Add the entire file structure"

→ "For each file, go through and list all the major entry points, describe what it does"

"Create Claude.md files for every major subfolder"

→ Claude will use the references made from the file structure and typically fetch the sub-folder Claude.md file (tip: add this suggestion to the Claude.md !)

Other .md files

- Use .md files to keep track of development notes
- Use Changelog.md to make it easy to keep track of major updates
- Use plan.md for large planning a large project.

We can ask him to update all this files without interfering with the main .md file

Claude Code in Github

/install-github-app

Cloud Code Commands

<https://claudecodecommands.directory/>

Claud self-corrects its own mistakes.

Types of Agents

- Chat-based (ChatGPT, Gemini, Claude Desktop, Super Grok)
- CLI/IDE Agent (Claude Code, Cursor, Windsurf, Kiro, etc)
- Background Agent (Codex, Jules, Devin etc)
- Agent Swarm (Factory Codex - in part, custom workflows)
- Non-engineering (Chat GPT Agent, Operator, Deep Research, Manus, etc)

What Agents Need (for Great Performance)


- Context
- Evals (examples, linters, standards, acceptance criteria, automated tests)
- Tools (MCPs, web search, bash, etc)

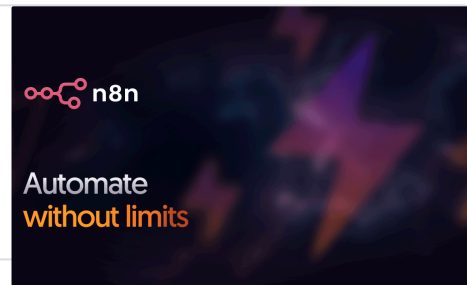
Beyond Engineering

- Second brain
- Computer Admin (naming screenshots, organizing files, pipe operator etc)
- Youtube Summary + execution tutorials
- Image manipulation, 3D modeling (Blender MCP)
- Building slide decks
- Workflow automations (e.g. n8n + headless Claude Code)

n8n.io - AI workflow automation tool

n8n is a free and source-available workflow automation tool

 https://n8n.io/?ps_partner_key=ZDQzZjVhNzkwNWlz&ps_xid=Jj9nQQbkdlM6o9&gsxid=Jj9nQQbkdlM6o9&gspk=ZDQzZjVhNzkwNWlz&gad_source=1&gad_campaignid=22924368945



Go-To MCP's

- Github (or via CLI)
- Playwright
- Context7
- FireCrawl
- Marketitdown
- Notion

The Three-Step Process

1. Explore
2. Plan
3. Execute

(Mandatory even on simple tasks)

If it views things wrong, clear the context!

→ Re-run everything!

/resume to resume from previous context

Risk-based Planning

Small, low risk:

→ Don't overthink - just write de code

Medium to large (>1 context window):

→ Ask Claude to break into testable, deployable PR's

High Risk Planning

→ Take 3 different shots at the plan

O Claude prefere criticar um Dev exterior do que a mim próprio, usar o truque do "Meu Dev Fez", como se algum outro dev exterior a nós tivesse feito as coisas e nós estamos a analisar com o Claude em duplas.

Large Project Power Moves

1. Get feedback on the plan
2. Ask another Claude for a lean plan with minimal edits
3. Ask a third Claude to decide between them
4. Save each to markdown
5. Have a 4th Claude review all plans

My Default Execute Prompt

"Now think hard and write elegant code that completes this.

Do not add backwards compatibility unless explicitly requested.

After every code block you write, lint, compile, and write corresponding tests and run them before writing the next code of block.

Select thinking mode

→ Think hard