> Debug Marketing

AI Agent Team Setup Guide

Bjorn Vanden Broeck
February 2026

No coding required. Built without a developer background.

> Debug Marketing

What This Guide Covers

AI agents can now work in teams. Not as a concept. As something I use every week to run my content engine.

This guide gives you the exact setup: what to install, how to get started, and how to build an AI team that handles your marketing content.

No coding required. I built this without a developer background. And I didn't write most of the technical setup myself — I asked Claude to do it.

What Changed: Opus 4.6

You probably use AI like this: type a prompt, get an answer, type the next prompt. One conversation, one task at a time.

Claude Opus 4.6 changed that. AI agents can now spawn teammates — each with their own role, their own context, working in parallel.

> **Old way:** You walk AI through every step. You are the project manager.

> **New way:** You describe the goal. AI assembles a team and figures out who does what.

Think of it like this: instead of managing one employee at a time, you now have a small department that coordinates itself.

The Team I Built

Here's the agent team I set up for my content engine:

Agent	Role	What It Does
Manager	Strategy guardian	Guards the monthly plan, decides what gets published
Planner	Content scout	Scans context files to find what's worth posting about
Post Creator	Writer + designer	Writes LinkedIn posts, generates hooks and carousel visuals
Newsletter Creator	Writer + designer	Writes the weekly Debug Marketing newsletter with visuals

Each agent knows its job. The manager keeps everything aligned with the monthly plan. The planner doesn't write — it researches. The creators don't plan — they execute.

The result? What used to take hours or days now takes minutes. A full week of content — posts, hooks, visuals, newsletters — generated in one session. Not random output. Strategic, consistent content that follows the plan.

What You Need

Requirement	Details
Subscription	Claude Max ($100/month or $200/month). Claude Pro ($20) does NOT include Claude Code.
VS Code	Free editor by Microsoft. This is where you'll work.
Operating system	macOS 13+, Linux (Ubuntu 20.04+), or Windows 10+ with WSL

This isn't free. $100/month is a real investment. For me, it replaced hours of manual work per week. That tradeoff was worth it. You'll have to decide that for yourself.

Setup: Step by Step

Step 1: Install Claude Code

Open your terminal (on Mac: search for "Terminal" in Spotlight) and run:

macOS / Linux:

```
npm install -g @anthropic-ai/claude-code
```

Windows (PowerShell):

```
npm install -g @anthropic-ai/claude-code
```

Then open VS Code and install the **Claude Code extension** from the Extensions marketplace. This is your main workspace. Everything happens inside VS Code from here.

Step 2: Open a project folder in VS Code

Create a new folder on your computer. Call it whatever makes sense: `marketing-system`, `content-engine`, `my-brand`. Open it in VS Code.

This empty folder is your starting point. You don't need to create any files yourself.

Step 3: Ask Claude to set up your project

Open Claude Code inside VS Code. Then just tell it what you want:

```
I want to build a marketing content system. I need a project
structure with a CLAUDE.md file, a context folder for my
background info, and an output folder for generated content.
Set it up for me.
```

Claude will create the folder structure, write a CLAUDE.md file (think of it as the project brain — it tells AI what this project is about), and set up everything.

You'll end up with something like:

```
my-marketing-system/
├── CLAUDE.md       # Project brain (Claude wrote this)
├── context/        # Background info for the AI
└── output/         # Where results go
```

Step 4: Tell Claude about yourself

Don't write context files by hand. Have a conversation instead:

```
I want you to create a context file about me so you can write
content in my voice. Ask me whatever you need to know.
```

Claude will interview you. It'll ask about your role, your experience, your audience, how you talk, what words you avoid. Then it creates the files based on your answers.

This is better than writing templates yourself because Claude structures the information in the way it can actually use it.

You can do the same for your audience:

```
Create a context file about my target audience. Interview me
about who I'm writing for, their pain points, and what kind of
content resonates with them.
```

Step 5: Start using it

Now you have a project with context. Ask Claude to do something real:

```
Plan 4 LinkedIn posts for this week based on my context files.
```

Or go bigger:

```
Research what kind of LinkedIn posts perform well in my niche,
then plan a week of content that matches my positioning and
audience.
```

The more context Claude has, the better the output. And the context gets better every time you work together. You'll naturally add more files, more rules, more examples over time.

Step 6: Sub-agents happen automatically

You don't create sub-agents (specialized AI workers that handle specific parts of a task) manually. Claude decides when to spawn them based on what you ask.

When you give Claude a task with multiple independent parts, you'll see something like:

```
Launching Explore agent to scan your context files...
Launching Explore agent to check recent post patterns...
```

These agents work in parallel. Each handles a focused subtask, then reports back to the main session.

To trigger sub-agent usage, ask for tasks with multiple parts:

- → "Scan my 3 project repos for updates AND plan this week's content based on what you find"
- → "Read my last 10 posts, identify what performed best, and suggest 5 new topics based on those patterns"
- → "Check my audience context, review my hook patterns, and generate 10 hooks for [topic]"

Single-step tasks ("Write a post about X") usually don't need sub-agents. That's fine. They're most useful for research and planning with multiple inputs.

How I Use This for Marketing

My content engine today

My content system has 47 context files across 10 directories. It didn't start that way. I started with 3 files and grew from there.

When Claude starts a session in my project, it knows:

- My positioning and what makes me different
- My 3 audience segments and what each cares about
- My content pillars and posting schedule
- 200+ hook patterns from analyzing successful creators
- 12 voice rules extracted from reviewing my own drafts
- 50 learnings from my projects that can become posts

I didn't write most of this manually. I told Claude what I needed, we had conversations, and it built the files. Then I reviewed and refined.

Example: weekly content planning

What I type:

```
Plan my content for this week. Check if any of the monthly
topics should be swapped based on what happened in my projects.
```

What happens:

1. Claude reads my monthly strategy (16 planned posts across 4 weeks)
2. Sub-agents scan my project repos for new developments
3. Another agent checks my learnings index for unused content ideas
4. Claude decides: keep 3 planned topics, swap 1 for something more timely
5. For each post: 5 hook options, a framework assignment, and an outline

Result: a concrete week plan with 4 posts ready to write.

Example: the full team in action

What I type:

```
Run the full content pipeline for this week.
```

What happens:

1. The **manager agent** checks the monthly plan and weekly rhythm
2. The **planner agent** scans context files and learnings for content opportunities
3. The **post creator agent** writes 4 LinkedIn posts with hooks, CTAs, and carousel visuals
4. The **newsletter creator agent** writes the weekly Debug Marketing newsletter
5. Everything follows the strategy — pillars, audience segments, voice rules

A full week of content in one session. Posts, visuals, newsletters — all aligned.

Example: repository scanning

What I type:

```
Scan my repos for new learnings
```

What happens:

1. Claude spawns one agent per repository (I monitor 5 across 2 GitHub accounts)
2. Each agent checks recent commits and changes
3. Results compiled into one report: what changed, what's worth a post

5 repos scanned in parallel instead of one by one.

Going Further: Custom Agents

Once you're comfortable, you can ask Claude to create specialized agents for you:

```
Create a custom agent called "content-reviewer" that only has
read access and reviews my posts against my voice profile.
It should use the faster Sonnet model to save costs.
```

Claude creates a file in `.claude/agents/` that defines this agent. From then on, you can call it by name:

```
Use the content-reviewer agent to check my latest post draft
```

Custom agents are optional. Sub-agents work without them. But they're useful when you want a consistent, reusable workflow that always checks the same things.

The Mindset That Makes This Work

The biggest shift isn't technical. It's how you think about AI.

> **Treat Claude like a new employee, not a tool.**

A new employee doesn't know your preferences on day one. They don't know your voice, your audience, or your standards. But they learn. Every time you give feedback, they get better.

Claude works the same way. The first output won't be perfect. That's fine. Give feedback. Be specific about what's wrong and what you'd do differently. Claude adjusts, and those adjustments stick in your project's context.

The feedback loop:

1. Ask Claude to do something (write a post, plan a week, research a topic)
2. Review the output. What's good? What's off?
3. Tell Claude what to change. "This sounds too formal." "I'd never say it this way." "The hook is too long."
4. Claude rewrites. Better this time.
5. Ask Claude to save what it learned as a rule in your context files

Over time, your project accumulates rules, preferences, and patterns. After a few weeks, Claude writes your first draft closer to how you'd write it yourself.

How to learn faster:

- → Watch videos and read guides about Claude Code best practices. There's a growing community sharing what works.
- → Then apply what you learned by asking Claude to implement it. "I saw that people use hook libraries for better posts. Can you research how to set one up and build it for this project?"
- → You don't need to understand the technical details. You need to understand what's possible, then ask Claude to make it real.

Test, fail, improve:

Don't aim for perfection. Aim for iteration.

- → Ask Claude to generate something. Review it. Give feedback.
- → Try a different approach. "What if we structured the post differently?"

- → When something works well, ask Claude to save that as a pattern for next time.
- → When something fails, tell Claude why. That feedback becomes part of your system.

The people who get the most out of this aren't the most technical. They're the ones willing to experiment, give honest feedback, and let the system learn from their input.

What I Learned (Honest Takes)

What works well:

- Parallel research saves real time. Scanning multiple sources simultaneously instead of sequentially.
- Context files compound. The more you add, the better every output gets. Not linearly — exponentially.
- Letting Claude build the structure is faster and better than doing it yourself. It knows what format it works best with.
- Feedback is the real skill. The better you get at explaining what's wrong, the faster the system improves.
- Agent teams change the workflow. Instead of managing each step, you describe the goal and let the team handle it.

What's still rough:

- Agent team capabilities are new. I'm still learning the boundaries.
- Not everything needs agents. Simple tasks are faster without the overhead.
- Context quality matters more than quantity. Bad context files produce bad outputs — just faster.
- The $100/month investment is real. It pays for itself if you use it daily, but it's not for everyone.

My advice: Don't try to set everything up perfectly on day one. Open VS Code, install Claude Code, and start a conversation. Ask it to help you build your system. It'll guide you through what it needs.

The power isn't in the agents themselves. It's in the context you build over time. And the best way to build it is together with the AI, one conversation at a time.

Follow Along

I'm documenting my entire journey building AI systems for marketing. Every win, every failure, every lesson.

Newsletter: Subscribe to Debug Marketing for weekly deep-dives on building AI systems for marketing — practical guides, not hype.

LinkedIn: Follow Bjorn Vanden Broeck for daily posts about what I'm building and learning.

February 2026. AI moves fast. The tools will change. The principle won't: give AI the right context and structure, and it becomes a team, not just an assistant.

Want to follow along?

I'm documenting my entire journey building AI systems for marketing. Every win, every failure, every lesson.

Subscribe to Debug Marketing →

Follow Bjorn Vanden Broeck on LinkedIn

> Debug Marketing

AI moves fast. The tools will change. The principle won't: give AI the right context and structure, and it becomes a team, not just an assistant.