Vibe coding for your *startup*

A hands-on workshop for the MBC Alumni Association

16 October 2026 · **Jonathan Engle**

The next 50 minutes

1. **Your marketing snapshot:** what you saw, how it's built, and how to fix it
2. **Vibe coding:** what it is and the words you'll hear
3. **Your company brain:** files that teach AI your business
4. **What you could build:** keys, examples and the build loop
5. **Shipping and teams:** putting it online and working together
6. **Setup, step by step:** the six steps you do once
7. **Working session:** start your own build

1

Your marketing snapshot

The Marketing Diagnostic Machine

Discovery and visibility 9

Paid ads · Category search · SEO · AI visibility · Local search · Brand search · Directories · YouTube · Social

Funnel and conversion 6

Speed to lead · Email sequences · Full funnel · Site technical · Landing pages · Retargeting pixels

Positioning and messaging 5

Message consistency · Competitor messaging · Reviews and customer voice · Case studies · AI-sounding writing

Operations 4

Marketing stack · Analytics · Attribution · CRM hygiene

Content and distribution 4

Distribution · Newsletter · PR and press · Community

Strategy 3

Competitive landscape · Market size · Keyword opportunity

Sales enablement 2

LinkedIn presence · Public sales collateral

33

Checks, each one an agent. Your snapshot ran the quick version of five areas.

How I built it

1

API keys

DataForSEO

2

An agent per check

33 markdown files, each with one job

3

An interpretation

scores each area and writes what to fix first

4

A client portal

the private link in your email

9

API keys behind the 33 checks

2 min 17 s

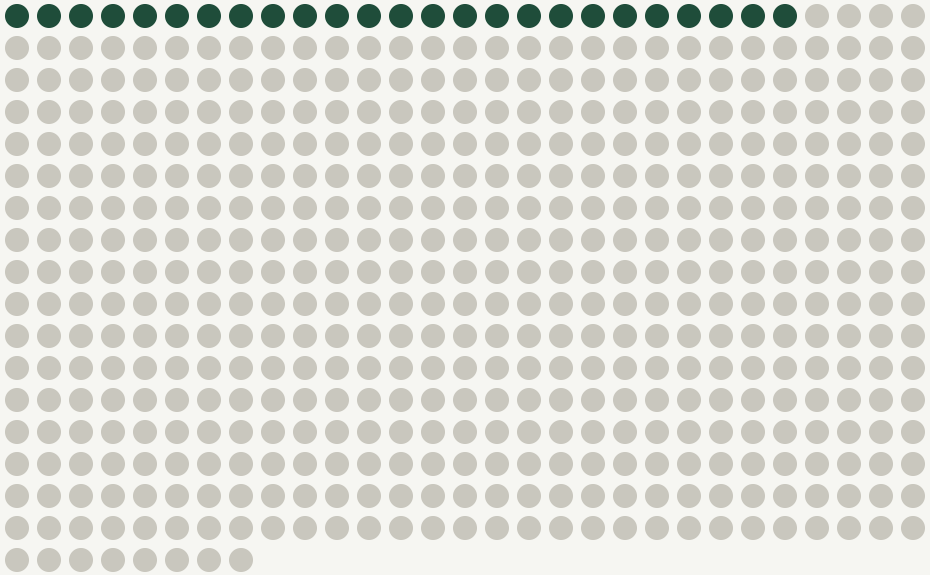
To measure one company, on the median run

$11.05

In AI and data costs to measure all 28 of you

I'm a marketer. I described each check in plain English, and Claude Code wrote the software.

This room's marketing snapshots



25 of 501 answers from ChatGPT, Gemini and Perplexity named one of your companies. Each dot is one answer.

28

Companies measured on search, AI answers, content, proof and site health

26 of 28

Missing from Google's top ten on every buyer search we checked

7.7 s

Median load time for a homepage's main content on a phone

Measured 13 to 15 October 2026. Your own snapshot, with what to fix first, is in the link emailed to you.

First builds from your snapshot

- **AI answers:** an FAQ page that answers the questions buyers ask ChatGPT about your category
- **Search:** one landing page for each buyer search you want to rank for
- **Site health:** have Claude find the heavy images and scripts slowing your homepage on a phone
- **Proof:** a customer stories page built from your best reviews
- **Content:** a weekly article draft built from your sales call notes

A first prompt

I run [company], which sells [what] to [who]. My marketing snapshot shows AI tools rarely name us. Build a one-page FAQ site that answers these six questions buyers ask: [paste them from your snapshot]. Plain HTML, mobile first, my brand colors [#hex]. Ask me anything you need to know before you start, then show me a plan.

Press `Shift+Tab` until you see plan mode, or type `/plan`. Claude writes the plan and changes nothing until you approve it.

2

Vibe coding

Vibe coding

YOU

Describe

what you want, in plain English

CLAUDE

Plan

the files and the steps

CLAUDE

Build

writes the code, runs it, fixes its own errors

YOU

Look

open the result and try it

YOU

Decide

ship it, or describe what's wrong

round again until it's right

Words you'll hear today

Terminal: a window where you type commands

Git: the tool that records that history

Commit: a saved snapshot of your project

API key: a password that lets your code use a service

Agent: written instructions Claude follows for one job, run when you ask

Repository (repo): a project folder that keeps its history

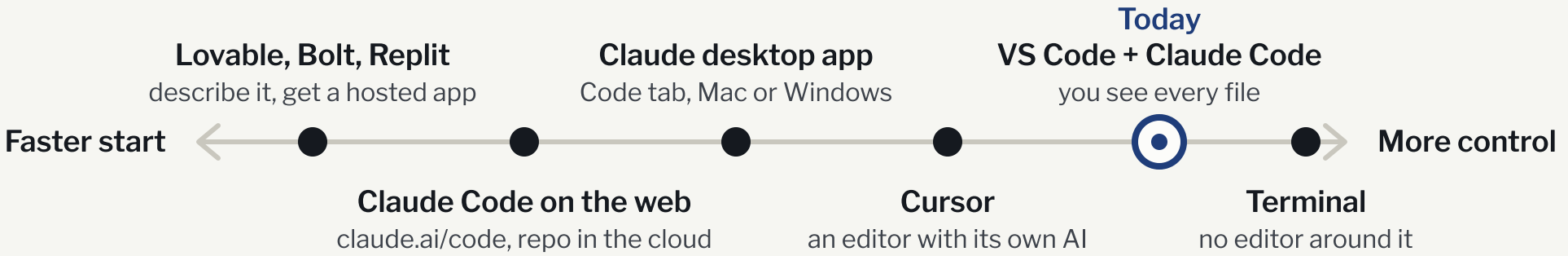
GitHub: the website that stores your repos online

Push: sending your commits up to GitHub

Deploy: putting your project on the internet

Routine: an agent on a schedule, which runs by itself with your laptop closed

Other ways to build



Describe, plan, look, fix works in every one of them.

3

Your company brain



Every prompt starts from zero, so everyone re-explains your company from memory.

The same question, twice

"Who do we sell to, and why do they pick us?"

Plain AI

Guesses from your website, if it knows you at all. Sounds like every company in your category.

AI with your brain

Answers from your own files, in your words, with only the facts you've approved.

The build order

4 **Agents** run your skills on a schedule or in Slack

3 **Skills** how your work gets done, step by step

2 **You** how you think, decide and write

1 **The brain** what's true about your company: customers, offer, voice, facts you may claim

What goes in your brain

```
brain/
  company.md      # what you sell, to whom
  customers.md    # who buys, in their words
  voice.md        # how you sound
  facts.md        # what you may claim
  me.md           # how you decide
  offer.md        # added after today
  competitors.md
  proof.md
```

- **Claude interviews you:** one question at a time, and writes the files
- **Your website goes in first:** Claude reads it, then asks about the gaps
- **Guesses get marked:** anything uncertain says [CHECK] until you confirm it
- **CLAUDE.md points at it:** every session reads the brain before writing

Four principles of AI readiness

Codified

Written down outside people's heads, in files a machine can read.

Current

One named person keeps it true, and drift shows up before a customer finds it.

Governed

A list says what you may claim, so nothing invents your results.

Connected

Plugged into the tools you already use, so nobody has to paste it in.

CLAUDE.md, the standing instructions

- **Read every session:** Claude opens it before it does anything
- **Started by /init:** Claude studies the project and writes the first version
- **Grown by you:** each time you correct Claude twice, add the rule

CLAUDE.MD

```
# FAQ page for Acme Dental

- Read every file in brain/ before writing
- Never state a fact not in brain/facts.md
- Write at an 8th-grade reading level
- Test on a phone-sized screen before
  saying a change is done
```

Agents are markdown files

- **A job description:** plain-English instructions for one recurring task
- **Kept in your repo:** in the .claude folder, so your whole team gets it
- **Called by name:** or picked up when Claude sees a matching job
- **Written by Claude:** "turn what we just did into an agent"

.CLAUDE/AGENTS/REVIEW-REPLIES.MD

```
---
name: review-replies
description: Drafts replies to new Google reviews
---
Read each new review in reviews.json.
Reply in our voice (see VOICE.md).
Thank them by name. Never offer a refund.
Save drafts in replies/ for a person to approve.
```

Three ways to run an agent

When you ask

You type one command. The Marketing Diagnostic Machine runs this way: one command measured this whole room.

On a schedule

A routine, set up with `/schedule`, runs in the cloud every Monday at 7am, with your laptop closed.

In Slack

Invite @Claude to a channel. Your team mentions it with a task and gets the answer, or a pull request, back in the thread.

A good first project

- **Weekly:** something you or your team already do by hand
- **Checkable:** you can tell in one look whether it's right
- **Small:** one page or one report, done in an afternoon
- **Public data first:** no customer records until you know where they'll be stored

4

What you could build



If you had this key, you could build this

Google Places	→	A reviews page that updates itself
PageSpeed Insights	→	An alert when your site slows down on phones
HubSpot or your CRM	→	Deals nobody has touched in two weeks
Calendly	→	A prep brief on each prospect before the call
Google Search Console	→	A Monday list of searches you almost rank for
OpenAI, Gemini, Perplexity	→	A weekly check of whether AI tools name you
Stripe	→	A revenue dashboard you don't rebuild by hand
Resend or Slack	→	Any of these, sent to your team every week

Every build follows four steps

1. **Get the key:** sign up for the service and create an API key
2. **Put it in .env:** one line per key, kept off GitHub by .gitignore
3. **Make the project:** a new folder in `~/dev`, opened in VS Code, with `git init`
4. **Kick it off:** one prompt in plan mode saying what to build, from what, and where the result goes

A new repo or a folder

Start a new repo

- **Ships alone:** it has its own site, link or schedule
- **Own keys:** it needs secrets nothing else should see
- **Own people:** a different team works on it

Add a folder

- **Part of a whole:** it's one piece of a bigger tool
- **Same brain:** it reads the same files and keys
- **Ships together:** it goes live with the rest

The Marketing Diagnostic Machine is one repo, with all 33 checks inside it. The portal you opened lives in my website's repo, because it ships with the site.

Example: an SEO agent

1. **Key:** sign up at dataforseo.com and add credit. You get an API login and password.
2. **.env:** the two lines on the right
3. **Project:** `mkdir ~/dev/seo-agent`, open it, `git init`
4. **Kickoff:** paste the prompt in plan mode

.ENV

```
DATAFORSEO_LOGIN=you@company.com
DATAFORSEO_PASSWORD=your-api-password
```

KICKOFF PROMPT

Build an SEO agent in this project. It reads DATAFORSEO_LOGIN and DATAFORSEO_PASSWORD from .env. For [yoursite.com], pull the searches we rank 4 to 20 for, and list the ten closest to page one with the page that should rank for each. Save the list in reports/ with today's date. Make it an agent I can run by name. Plan first.

Example: a reviews page that updates itself

1. **Key:** in Google Cloud, turn on the Places API and create an API key
2. **.env:** the one line on the right
3. **Project:** `mkdir ~/dev/reviews-page`, open it, `git init`
4. **Kickoff:** paste the prompt in plan mode

.ENV

```
GOOGLE_PLACES_KEY=your-api-key
```

KICKOFF PROMPT

Build a reviews page for [company] that pulls our newest Google reviews with the Places API, using GOOGLE_PLACES_KEY from .env. Find our listing by name and address first and show me what you found. One page of plain HTML, mobile first, in our colors [#hex]. Rebuild it each time I run it. Plan first.

Example: a revenue dashboard

1. **Key:** in Stripe, create a restricted key with read-only access
2. **.env:** the one line on the right
3. **Project:** `mkdir ~/dev/revenue`, open it, `git init`
4. **Kickoff:** paste the prompt in plan mode

.ENV

```
STRIPE_KEY=rk_live_your-restricted-key
```

KICKOFF PROMPT

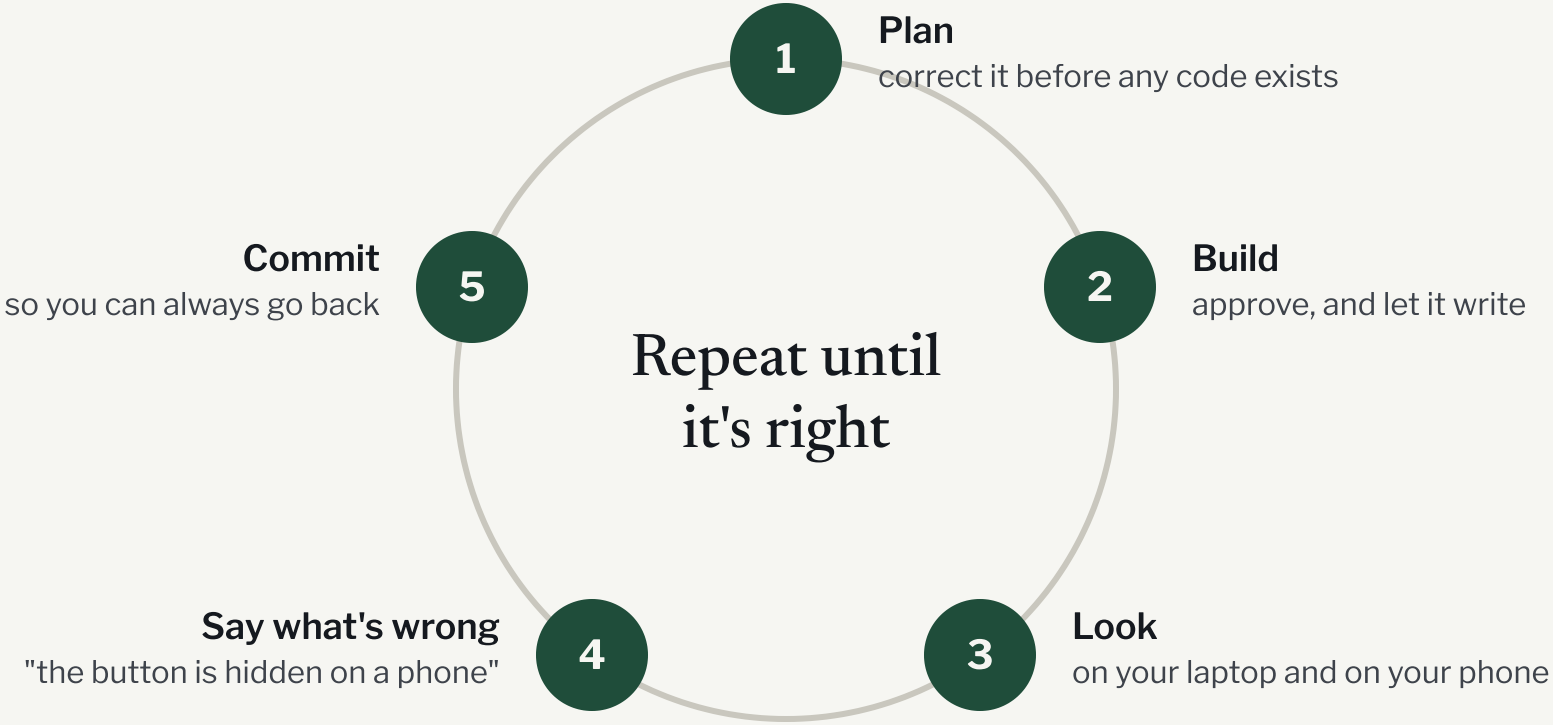
> Build a revenue dashboard from Stripe using STRIPE_KEY in .env. The key is read-only, and nothing you build may write to Stripe. Show monthly revenue for the last 12 months, new customers by month, and refunds. One HTML page I open on my laptop, refreshed each time I run it. Plan first.

Questions for your table

1. **What do you copy and paste** between two tools every week?
2. **Which report** do you rebuild by hand?
3. **What would you send every customer** if it took no time?
4. **Which number in your snapshot** bothers you most?

When this happens, pull this data from this tool, and send this result to this person.

The build loop



5

Shipping and working as a team

Putting it online with Netlify

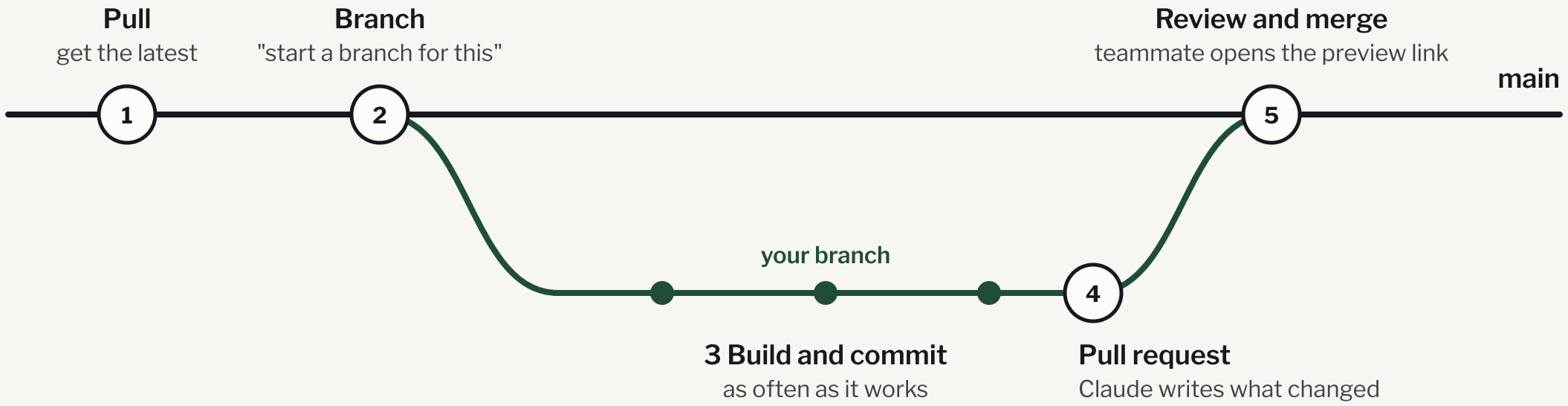
Today

1. Go to **app.netlify.com/drop**
2. **Drag your site folder** onto the page, never the folder holding .env
3. **Open the link** it gives you on your phone

From then on

1. **Connect the GitHub repo** in Netlify
2. **Push a change**, and the site updates itself
3. **Add your domain** when it's ready

A day of building with a teammate



Roles on an AI-built project

Builder

Drives Claude, owns the plan, and commits the work.

Reviewer

Opens the preview link Netlify makes for each pull request, and says what's wrong. No setup needed.

Requester

Asks for changes in Slack or the web app, and uses the agents the builder writes.

One person can hold all three. Name who holds which before the first build.

Team rules

- **One shared repo:** invite teammates in GitHub under Settings, then Collaborators
- **One CLAUDE.md:** everyone's Claude follows the same rules
- **Shared agents:** a good agent goes in the repo, so the whole team gets it on their next pull
- **Own .env each:** share keys through a password manager, never Slack or email
- **A data rule, set first:** decide what customer data the AI may see before anyone pastes any
- **Main always works:** nothing merges until someone has opened it and tried it

Habits that make you faster

One job

One task per session. Type `/clear` when the task changes.

Plan first

A wrong plan costs a minute. Wrong code costs an hour.

Show it

Paste a screenshot of what looks wrong.

Check it

"Open the page at phone width and confirm the button works."

Talk it

Dictate long prompts. You'll give more detail.

Automate it

The second time you do something by hand, have Claude script it.

The /retro habit

1. **Finish a working session**
2. **Type /retro**
3. **Claude lists** what slowed it down and what worked
4. **It writes the fix** into CLAUDE.md, a skill or a script, and tests it
5. **Your next session** starts with the lesson

.CLAUDE/SKILLS/RETRO/SKILL.MD

```
---
name: retro
description: Reflect on this session,
  then fix what slowed it down
---
List the mistakes and repeated steps.
For each, write the rule or script that
prevents it, and prove it works.
```

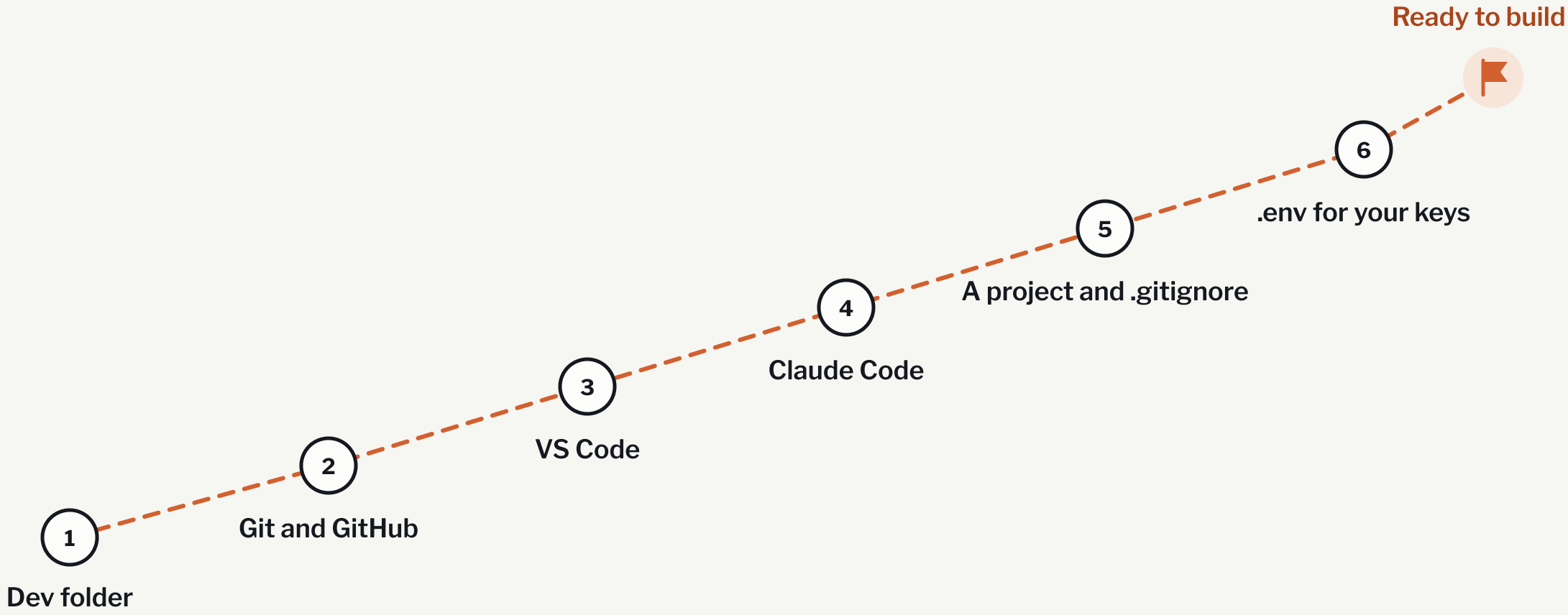
Your plan for next week

1. **Open your snapshot** and find your lowest-scoring area
2. **Finish today's build,** or start one for that area
3. **Put it online** with Netlify
4. **Send the link** to one person in this room
5. **Run /retro** at the end of each session

6

Setup, step by step

Setup in six steps



Step 1: a dev folder

- **One home:** every project lives in one folder
- **Not synced:** keep it out of Dropbox, OneDrive and iCloud, which can corrupt a project's history
- **Backed up:** GitHub becomes the backup in step 2

MAC: TERMINAL · WINDOWS: POWERSHELL

```
mkdir ~/dev
# makes a folder called dev in your home folder
```

Step 2: Git and GitHub

1. **Sign up** at github.com/signup
2. **Install Git:** on a Mac, run the command at right and accept the install; on Windows, download it from git-scm.com
3. **Keep repos private** until you choose otherwise

CHECK THAT GIT IS INSTALLED

```
git --version
# prints a version number when it's ready
```

Steps 3 and 4: VS Code and Claude Code

1. **Install VS Code** from code.visualstudio.com (free)
2. **Open Extensions:** `Cmd+Shift+X` on a Mac, `Ctrl+Shift+X` on Windows
3. **Search "Claude Code"** and install the one published by Anthropic
4. **Sign in** with a paid Claude plan (Pro or higher; the free plan doesn't include Claude Code)
5. **Trust the folder** when VS Code asks, or the extension won't run



Step 5: a project and its .gitignore

MAKE THE FOLDER, THEN OPEN IT IN VS CODE

```
mkdir ~/dev/faq-page
```

THEN ASK CLAUDE

Set this folder up as a git repo with a .gitignore that keeps out .env files, node_modules and system files. Then make it a private GitHub repo.

WHAT THE .GITIGNORE HOLDS

```
.env
.env.*
!.env.example
node_modules/
.DS_Store
# files git will never upload
```

Step 6: a .env file for your keys

- **Keys live in .env:** never in your code, a chat or an email
- **.env stays home:** the .gitignore keeps it off GitHub
- **.env.example travels:** same names, no values, so a teammate knows what to fill in
- **A leaked key gets replaced:** make a new one at the provider and delete the old

.ENV (PRIVATE)

```
OPENAI_API_KEY=sk-proj-8f2 ...
RESEND_API_KEY=re_4kq ...
```

.ENV.EXAMPLE (SHARED)

```
OPENAI_API_KEY=
RESEND_API_KEY=
```

Command cheat sheet

IN THE TERMINAL

```
mkdir ~/dev          # make your dev folder
git --version        # check Git is installed
git pull             # get your team's changes
git status           # see what's changed
```

IN CLAUDE CODE

```
/init                # write a CLAUDE.md
/plan                # plan before building
/clear               # start a fresh task
Shift+Tab            # switch modes
"commit this"        # save a snapshot
"push to GitHub"     # back it up online
```

16 min

Working session

Every prompt: **thejonathanengle.com/vibe-coding**

1. **Start your brain:** paste the brain prompt and answer Claude's questions
2. **Point CLAUDE.md at it**
3. **Ask the same question twice:** Claude, then ChatGPT
4. **Paste your one sentence** in plan mode, approve, build
5. **Open the result** on your phone

Stuck? Paste the error into Claude first, then ask your neighbor, then raise your hand.

Build one thing this *week*

Set up.

Six steps, done once.

Build.

Plan, build, look, fix.

Ship.

A link someone can open.

The guide: thejonathanengle.com/vibe-coding

Jonathan Engle · calendly.com/jonathanengle