Website-First Outreach Setup

Set up the Claude skill that finds local businesses with dated websites, builds each one a replacement, and delivers it through their own contact form.

Why I stopped sending cold emails

A cold email asks a business owner to imagine what you could do for them. They have to read it, believe it, and then decide to reply. Almost nobody makes it past step one, which is why the reply rate on a normal outreach list sits close to zero.

This system changes what arrives in their inbox. Instead of a pitch, they get a working preview of their own business website, already built, already hosted, already live at a link they can open on their phone. The question stops being whether you can do the work, because the work is sitting in front of them.

The whole thing runs as an agent skill called local-business-revamp. It is open source, it runs on your own machine, and Claude Code or OpenAI Codex drives it. You give it an industry, a city, and a number. It handles discovery, qualification, the build, the hosting, and the send. This guide is how to get it running.

01

How it actually works

Each prospect moves through a fixed set of stages, and the skill writes every stage into a local SQLite ledger so nothing gets lost and nobody gets contacted twice.

1. Discovery and qualification. The agent searches your industry and city, opens each business's real website, and judges it only on how it looks: fixed-width layouts, tiny navigation, illegible type, stretched images, a mobile view that falls apart. It also screens the contact page and confirms there is a genuine enquiry form with a message field and a submit button, saving that URL as contact_url.
2. Build and deploy. The agent collects the business name, public phone, email, and listed address from the site itself, picks a theme colour from the business's own branding, and runs the generator against a prepared template. The output is a static folder under runs/quick/ that goes straight to its own isolated Vercel project. Then the agent opens the public URL in a logged-out context to confirm a stranger can load it.
3. Contact and record. The agent fills the business's original contact form with a fixed message carrying the verified preview link, reserves the attempt in the ledger with contact-begin, clicks submit once, and records what actually happened with contact-finish. No retries, no follow-ups, no switching to email.

The full stage path is queued, reviewing, extracting, building, checking, deploying, ready, contacting, then one of complete, sent, uncertain, blocked or skipped. If you ever wonder where a job is, that list is the answer.

02

The one thing almost everyone gets wrong

People qualify prospects by how old the website looks on paper. They check the copyright year in the footer, or the domain registration date, and treat anything before 2018 as a lead. The skill explicitly forbids this, and the reason is that the signal lies in both directions.

A plumber who updated their footer last month can still be running a layout from 2011. A site with a stale copyright line can be clean, fast, and perfectly readable on a phone. Send the second one a rebuilt website and you look careless. Skip the first one and you miss the best prospect on the page.

So qualify on what you can see. Open the site on desktop, open it narrow, and write down at least two concrete visual problems before the job moves forward. Those findings stay internal. They never go in the message, because opening with what is wrong with someone's website turns a gift into a critique.

> **WATCH OUT**
>
> Screen the contact form BEFORE you build, not after. An email address, a phone number, a newsletter signup or a review form does not count. You need a real enquiry form with a message field and a submit control. Building a site for a business you cannot reach wastes the build, and the skill will route that job to manual handoff anyway. Recheck the saved form again immediately before sending, because forms change.

---

03 The files

The skill folder is the application root. Everything below lives inside it, and the paths in SKILL.md resolve relative to it.

> **SOURCE CODE**
>
> The full repo, including the SKILL.md, the generator, the ledger CLI and the dashboard:
>
> **https://github.com/albertshiney/websitegenerator**

Worth knowing what is in there: SKILL.md is the workflow the agent follows. tools/quick_site.py is the generator. tools/control.py is the ledger CLI. tools/scout_screen.py and tools/site_assets.py handle the Firecrawl screening. dashboard/ is the local contact sheet. templates/electrician/ is the prepared site design. references/ holds the outreach, coordination and production rules. data/revamp.sqlite3 is the ledger, and runs/ holds the per-business artifacts.

.ENV

```
# Copy from .env.example. settings.json points firecrawl_env_file at this file.

# Firecrawl API key for homepage and contact-page screening (tools/site_assets.py).
# Required for scouting.
FIRECRAWL_API_KEY=

# Optional: path to the SQLite ledger. Defaults to data/revamp.sqlite3.
# Leave commented unless overriding. An empty value breaks the default.
# REVAMP_DB=

# Optional: Chrome or Chromium binary for the template prerender script.
# Defaults to the macOS Google Chrome path.
CHROME_PATH=

# Optional: port for the prerender script's static server. Empty picks a free port.
PRERENDER_PORT=
```

SETTINGS.JSON (THE FIELDS YOU ACTUALLY CHANGE)

```
{
  "dashboard_port": 4310,
  "batch_size": 5,
  "requested_workers": 5,
  "vercel_scope": "your-vercel-team-slug",
  "vercel_cli": ["pnpm", "dlx", "vercel@59.16.0"],
  "firecrawl_env_file": ".env",
  "max_scrape_pages_per_business": 5,
  "max_discovery_candidates": 20,
  "sender": {
    "name": "Your Name",
    "phone": "+44 20 7946 0958",
    "email": "you@yourdomain.com",
    "first_name": "Your",
    "last_name": "Name"
  },
  "default_template": "electrician",
  "personalization_fields": [
    "name", "phone", "email", "street", "city", "zip", "theme_color"
  ]
}
```

Those seven personalization fields are the only things that change per business. The copy, the layout and the stock photography stay exactly as the template ships them, which is what makes a batch fast. A single generation takes about 30 seconds.

GENERATE ONE SITE BY HAND

```
./website --theme-color "#2563eb" \
  --name "Business Name" \
  --phone "+44 20 7946 0958" \
  --email "hello@business.com" \
  --street "12 High Street" \
  --city "Boston" \
  --zip "02108"

# ./website is a one-line wrapper around:
#   python3 tools/quick_site.py create
# Output lands in runs/quick/<id>/site, which is the deployable folder.
```

THE FIXED OUTREACH MESSAGE (DO NOT REWRITE IT PER PROSPECT)

```
Hey!

Your website is decent, but looks a little old.
I made an updated one.

You can take a look here: [VERIFIED PREVIEW URL]

The website is totally free, you just pay for hosting.

If you'd like help setting it up, text me on [YOUR PHONE] or email [YOUR EMAIL].

Best,
[YOUR FIRST NAME]
```

Same wording every time. Only the preview URL and the business name in the optional subject line change. No SEO claims, no "I checked your website", no custom praise. Spinning the message per prospect is how this turns into spam.

Set it up in Claude Code or Codex

Budget about twenty minutes for the first run, most of which is waiting on account logins.

1. Clone the repo into your skills folder. For Claude Code: git clone https://github.com/albertshiney/websitegenerator ~/.claude/skills/local-business-revamp. The folder name has to match the name: field at the top of SKILL.md, which is local-business-revamp. If those two disagree, the skill will not load.
2. For Codex instead, clone it into your project folder or into .codex/skills/local-business-revamp, then add a line to AGENTS.md pointing at the SKILL.md path so the agent finds it.
3. Check Python 3 is there. Run python3 --version inside the folder. That is all the generator needs. Node and npm are only required if you want to edit the master template, so skip them for now.
4. Get a Firecrawl API key and put it in .env. Run cp .env.example .env, then paste the key after FIRECRAWL_API_KEY=. Ask Claude to do this for you if you would rather not open the file. Leave the other three variables alone unless you have a reason.
5. Fill in your sender details in settings.json. The sender block is what gets typed into every contact form: your name, phone, email, first name and surname. Set vercel_scope to your own Vercel team slug. Get this wrong and you will be sending someone else's contact details to strangers.
6. Log in to Vercel and confirm the scope. Run pnpm dlx vercel@59.16.0 whoami and then pnpm dlx vercel@59.16.0 teams ls. Both are read-only. You do not need the global vercel binary installed. If whoami fails, log in first and come back.
7. Start the dashboard. Run python3 tools/control.py serve --port 4310 and open the localhost URL it prints. The dashboard is a read-mostly contact sheet showing live progress and previews. It does not launch the agent, so do not sit there waiting for it to do something.
8. Run your first batch as a dry run. Tell the agent to qualify two businesses in one city and stop before sending anything. This proves discovery, screening and the build work before a single message leaves your machine.
9. Check the ledger. Run python3 tools/control.py state. A healthy first run shows a batch with two jobs, each carrying a preview_url, preview_verified set to true, and a contact_url. That is what success looks like before you turn sending on.

> 💡 **QUICK TIPS**
>
> - The agent needs a browser surface to inspect sites and fill forms. In Codex that is the in-app browser. If a worker has no browser, the coordinator takes over those steps rather than marking the job blocked.
> - Back up the data/ folder before you move the application anywhere. That SQLite file is the only record of who you have already contacted.
> - Batches default to five businesses and accept anything from 1 to 100. Start at two. Read the messages that go out. Then scale.
> - Never edit .env or settings.json values into a chat message or a screenshot. Keep the Firecrawl key out of anything you share.

---

04 The commands I actually run

These are prompts you give the agent, not shell commands. Swap the bracketed parts for your own values.

THE DRY RUN, EVERY SINGLE TIME BEFORE A REAL BATCH

```
Use the local-business-revamp skill. Dry run only: find and qualify [2] [INDUSTRY] businesses in [CITY]. Build and deploy their previews and verify the public URLs, but do NOT submit anything to any contact form. Show me each preview link and the two visual findings you qualified on.
```

THE REAL BATCH

```
Use the local-business-revamp skill. Run a real batch of [5] [INDUSTRY] in [CITY]. Carry every qualified prospect all the way through: qualify, screen the contact form, build, deploy, verify the public URL logged out, then submit the fixed message once. Report the counts at the end: requested, reviewed, hosted, submitted, manual, skipped, blocked.
```

CHECK WHERE EVERYTHING STANDS

```
Run python3 tools/control.py state for the local-business-revamp skill and tell me, for batch [BATCH_ID]: completed_count, remaining_count, active_count, and every job that is not terminal. For each unfinished job, tell me the stage it is stuck in and why.
```

BUILD ONE SITE BY HAND FOR A BUSINESS YOU ALREADY KNOW

```
Use the local-business-revamp generator. Build a site for [BUSINESS NAME], phone [PHONE], email [EMAIL], address [STREET], [CITY], [ZIP]. Pick the theme colour from their existing logo and tell me the hex you chose and why. Deploy it to its own Vercel project and give me the public URL, then confirm you opened it logged out.
```

PICK UP AN INTERRUPTED RUN

```
Use the local-business-revamp skill. Before creating anything new, run state and resume the unfinished jobs in batch [BATCH_ID]. Recheck the evidence on each one from scratch. Do not recover any job that has a reserved contact attempt, and do not re-contact anything already marked complete or sent.
```

When it breaks

Every one of these has bitten a real run. The fix column is the actual fix, not a suggestion to try again.

ERROR	FIX
Choose an available template: electrician	You passed a --template value the repo does not have. Only the electrician template ships. Add your own under templates/, register it in settings.json, and keep the same seven personalization fields.
Theme color must be a six-digit hex color, e.g. #2563eb	Three-digit shorthand and named colours are rejected. Use the full six digits with the leading #. If the business has no clear branding, fall back to the electrician blue #2563eb and record that it was a fallback.
Provide a phone number or email address	The generator refuses a business with neither. Go back to the original site and find a public contact method. If there genuinely is not one, skip the prospect. A site with no way to reach anyone is not worth building.
The preview URL asks a visitor to log in to Vercel	The project was deployed under a protected scope. Confirm your scope with pnpm dlx vercel@59.16.0 whoami, redeploy to an isolated public project, and open the URL in a private window before you put it in any message. An unreachable preview blocks outreach.
add failed with a uniqueness error	Another job in the ledger already owns that domain, phone or email. Inspect the existing job. Do not change the key to force the insert, because that is how you contact the same business twice.
contact-begin was rejected	The reservation refuses duplicates, missing qualification evidence, failed QA and unverified previews. Fix the underlying job first: set qa_passed true after real checks, set preview_verified true only after loading the public URL. Never submit the form when the reservation failed.
A CAPTCHA appeared on the contact form	Leave it alone. Finish the website, verify the preview, then run manual-outreach so the job lands in the manual bucket with the message and contact details prepared. Do not solve it and do not fall back to email or SMS.
Firecrawl screening fails or returns nothing	FIRECRAWL_API_KEY is missing or empty in .env, or settings.json is pointing firecrawl_env_file somewhere else. Check both. Never print the key value into a log, a chat message or the dashboard.