DeltaForceOS

CODEX ROUTE / BEGINNER MODULE 05

Build Your First $5 Agent: Codex Route

BUILD OUTCOME

Ship a small but real agent that costs almost nothing and teaches the full loop: input, model, output, log, review.

BEST FOR

Use this PDF when you want exact prompts and clicks for this route instead of general theory.

MODEL TIER

Light by default; mid for prioritization; heavy only for complex synthesis.

FIRST INSTRUCTION

```
Inspect this repo for the Build Your First $5 Agent build.
Explain the files a beginner needs: README.md, package.json, src, public, scripts, .env.local, and any
Supabase files.
Do not edit yet. Give me the smallest safe build plan.
```

Before using the agent, make the workspace boring and clear.

01 Open the project folder. If you do not have one yet, create a new folder on Desktop and name it after the lesson.

02 Install Node.js if npm is not available. Beginners can check by opening Terminal and typing npm -v.

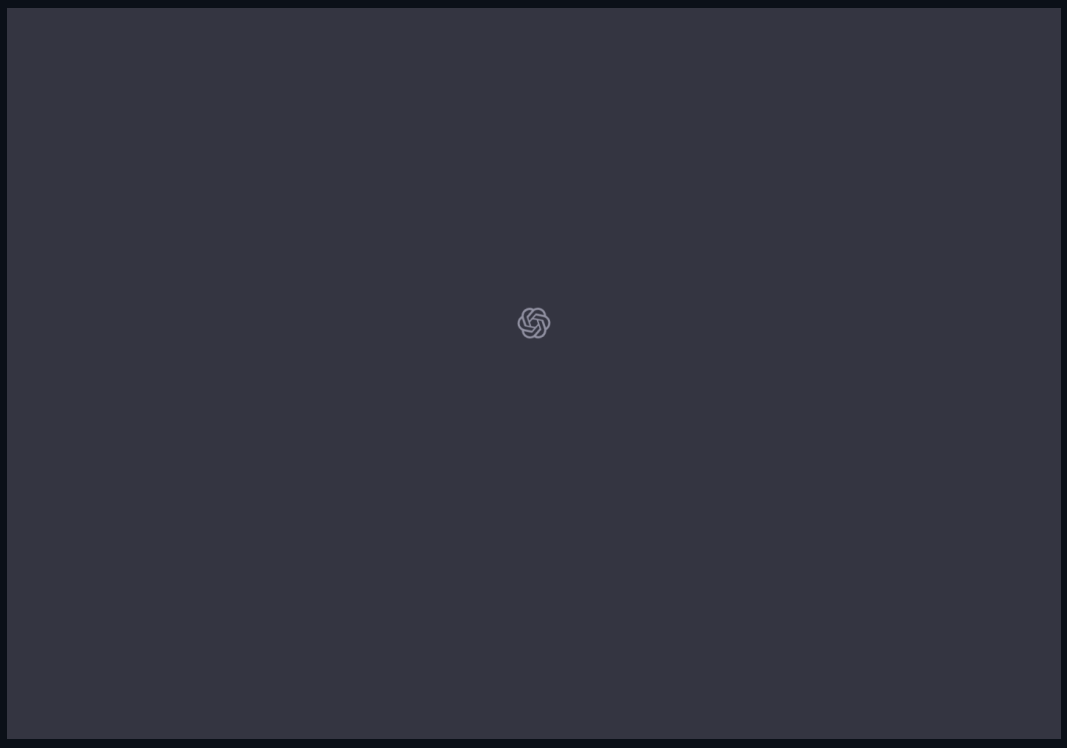
03 Create .env.local in the project root and add only the keys this lesson needs.

04 Confirm .env.local is listed in .gitignore so secrets do not go to GitHub.

05 Open the official dashboard for the first two tools in this lesson and create one tiny test project.

06 Decide where proof will live: Supabase row, screenshot, live URL, or Skool post.

OpenAI: click path for this route.



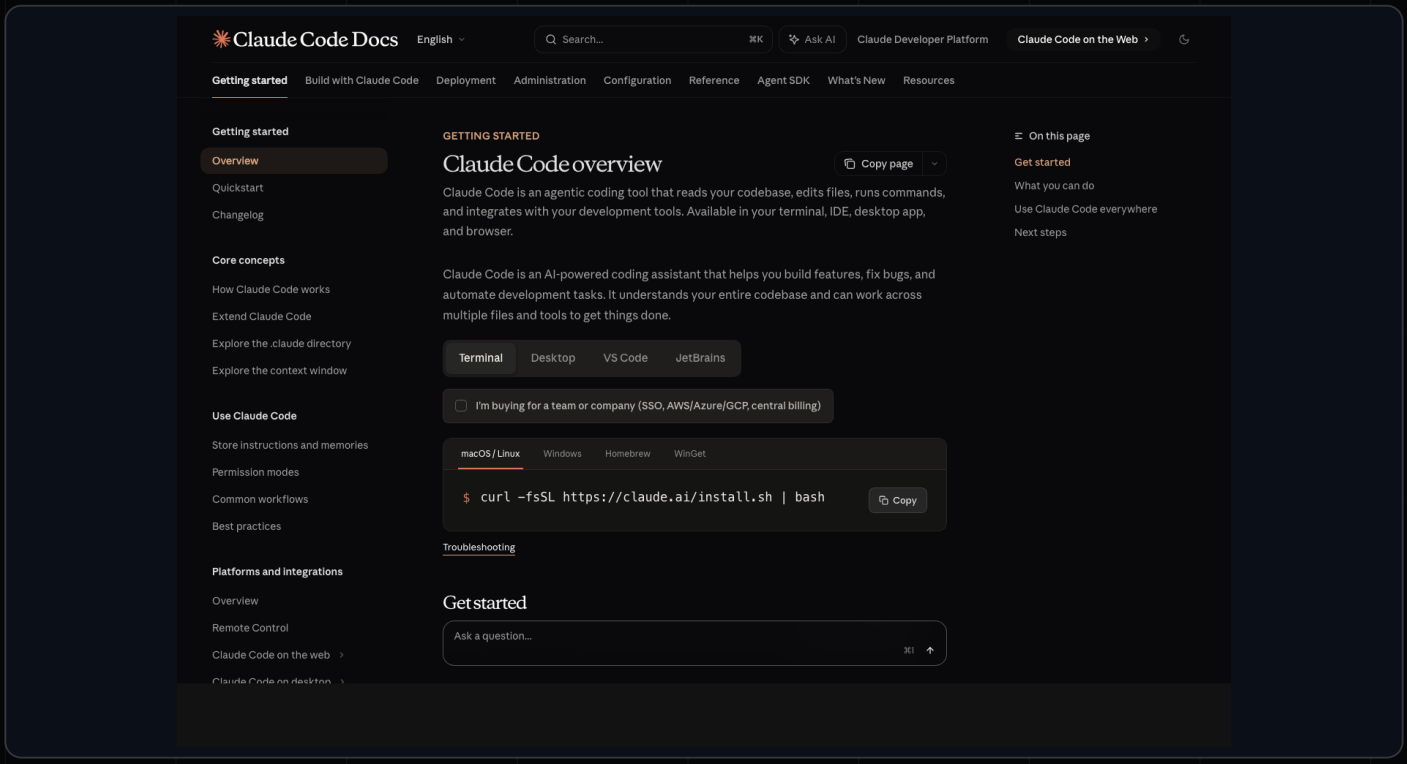
WHERE TO CLICK

Open the OpenAI platform, create an API key, add billing limits, and start with one small Responses API call.

ENV NAMES

```
OPENAI_API_KEY
```

Claude: click path for this route.



WHERE TO CLICK

Open Anthropic Console, create a key, and use Claude for planning, review, and long-context reasoning.

ENV NAMES

```
ANTHROPIC_API_KEY
```

Copy this box exactly, then let the agent ask clarifying questions only if needed.

INSPECT PROMPT

```
Inspect this repo for the Build Your First $5 Agent build.
Explain the files a beginner needs: README.md, package.json, src, public, scripts, .env.local, and any
Supabase files.
Do not edit yet. Give me the smallest safe build plan.
```

BEGINNER CHECK

If the agent proposes a huge rebuild, ask it to shrink the scope to the smallest artifact that proves this lesson.

DO NOT PASTE

Do not paste service-role keys, payment secrets, signing keys, or private client data into the chat.

Copy this box exactly, then let the agent ask clarifying questions only if needed.

BUILD PROMPT

```
Implement the smallest useful version of Build Your First $5 Agent.
Outcome: Ship a small but real agent that costs almost nothing and teaches the full loop: input, model,
output, log, review.
Tools allowed: OpenAI, Claude, Supabase, Vercel, Resend, PostHog, Python, TypeScript
Keep secrets server-side. Add code comments only where a beginner would get lost.
```

BEGINNER CHECK

If the agent proposes a huge rebuild, ask it to shrink the scope to the smallest artifact that proves this lesson.

DO NOT PASTE

Do not paste service-role keys, payment secrets, signing keys, or private client data into the chat.

TYPE THIS INTO FILES

```
# README.md lesson block
Lesson: Build Your First $5 Agent
Outcome: Ship a small but real agent that costs almost nothing and teaches the full loop: input, model,
output, log, review.
Tools: OpenAI, Claude, Supabase, Vercel, Resend, PostHog, Python, TypeScript

# .env.local placeholders
NEXT_PUBLIC_SUPABASE_URL=your_value_here
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_value_here
OPENAI_API_KEY=server_only
ANTHROPIC_API_KEY=server_only

# Verify
npm run build
```

WHERE IT GOES

README.md gets instructions. .env.local gets real local values. Vercel gets production values. Source files get implementation.

Basic English. Click by click. Do not skip ahead.

01 Open your operator project from Beginner 04 in your editor.

02 Open chrome and go to platform.openai.com. Sign in. Click API Keys in the left sidebar. Click Create new secret key. Name it operator. Copy the key into a sticky note.

03 Add OPENAI_API_KEY=your-key to .env.local. Save the file. Never commit .env.local to GitHub.

04 Open Terminal in your project folder. Type npm install openai @supabase/supabase-js and press Enter.

05 In your editor, create a new file at src/app/api/brief/route.ts.

06 Paste this starter code into route.ts: import OpenAI from 'openai'; export async function GET() { const ai = new OpenAI(); const r = await ai.chat.completions.create({ model: 'gpt-4o-mini', messages: [{ role: 'user', content: 'Give me 3 short news ideas for AI builders today.' }] }); return Response.json({ output: r.choices[0].message.content }); }

Finish the build. The last step always posts proof in Skool.

01 Save the file. With npm run dev still running, open chrome to http://localhost:3000/api/brief. You should see JSON with three ideas.

02 If you see a 401 error, your key is wrong. Open .env.local and confirm the key has no quotes and no extra spaces. Restart npm run dev.

03 Open supabase.com, click your operator project, click Table editor in the left sidebar.

04 Click + New table. Name it agent_runs. Add columns: id (uuid, primary, default uuid_generate_v4()), output (text), cost (numeric), created_at (timestamp, default now()). Click Save.

05 Add a Supabase insert at the end of route.ts: import the createClient helper, call .from('agent_runs').insert({ output: r.choices[0].message.content }). Save the file.

06 Refresh /api/brief twice in chrome. Open Supabase Table editor and confirm two rows in agent_runs. Screenshot both rows for the Beginner 05 Skool thread.

The build is not finished until verification is boring.

01 Does the agent stay under the weekly budget?

02 Does it produce fewer than five ranked actions?

03 Can you trace a bad answer back to its source input?

04 Run npm run build or the focused test command.

05 Download or open the produced artifact and inspect it visually.

06 Write the failure and fix in README.md or the Skool proof post.

01 Commit safe files to GitHub after reviewing the diff.

02 Open Vercel and deploy a preview first.

03 Add missing env vars under Project Settings -> Environment Variables if the build fails.

04 Run the workflow with fake or test data in preview.

05 Promote or deploy production only after the smoke test passes.

06 Save the live URL and proof artifact in Skool.

AGENT DEPLOY PROMPT

```
Verify the preview build, list any missing env vars, and give me the exact smoke test before production.
```

If something goes wrong, ask for a small repair instead of a total rewrite.

01 Making the first agent too broad to judge.

02 Skipping cost limits because early usage feels tiny.

03 Letting the model overwrite state instead of appending history.

04 Ask the agent to explain the failing layer: local app, env var, database, external API, prompt, deploy, or auth.

05 Ask for one fix, one test, and one rollback step.

06 Do not let the agent change unrelated files to hide the failure.

SKOOL PROOF

Build the smallest daily agent you would still want to run for seven days.

WHAT TO POST

Artifact, screenshot or URL, what worked, what broke, exact prompt used, and the next target.

SOURCES

Codex: https://developers.openai.com/codex

OpenAI: https://platform.openai.com/docs/quickstart

Claude: https://docs.anthropic.com/en/docs/overview

Supabase: https://supabase.com/docs/guides/getting-started

Vercel: https://vercel.com/docs/environment-variables

Resend: https://resend.com/docs/send-with-vercel-functions