DeltaForceOS

CODEX ROUTE / BEGINNER MODULE 04

Daily Operator Setup: Codex Route

BUILD OUTCOME

Have a clean workspace that can safely run your first real automation every morning.

BEST FOR

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

MODEL TIER

Light for daily summaries; mid if the brief must rank tradeoffs.

FIRST INSTRUCTION

```
Inspect this repo for the Daily Operator Setup 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.

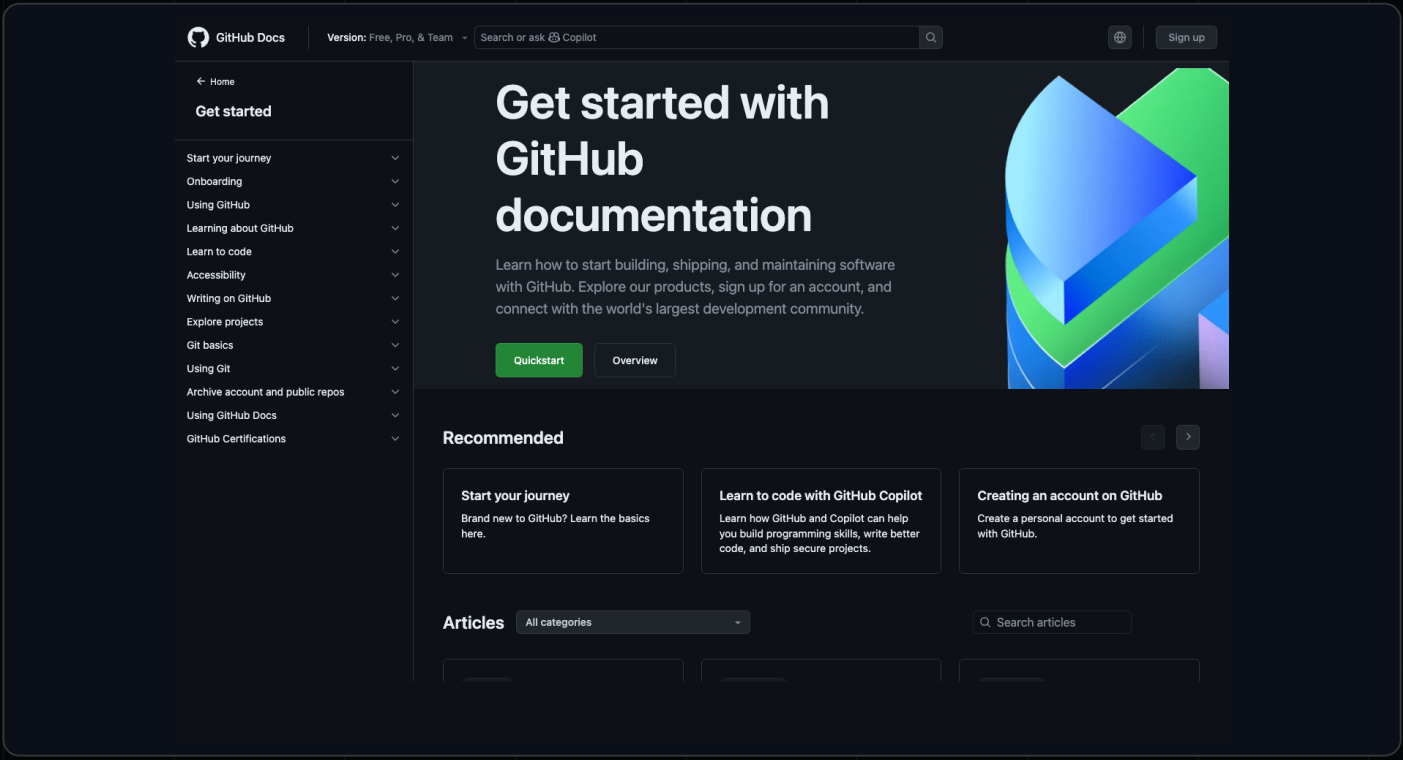
03 Create .env.local in the project root and add only the keys this lesson needs.

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.

GitHub: click path for this route.



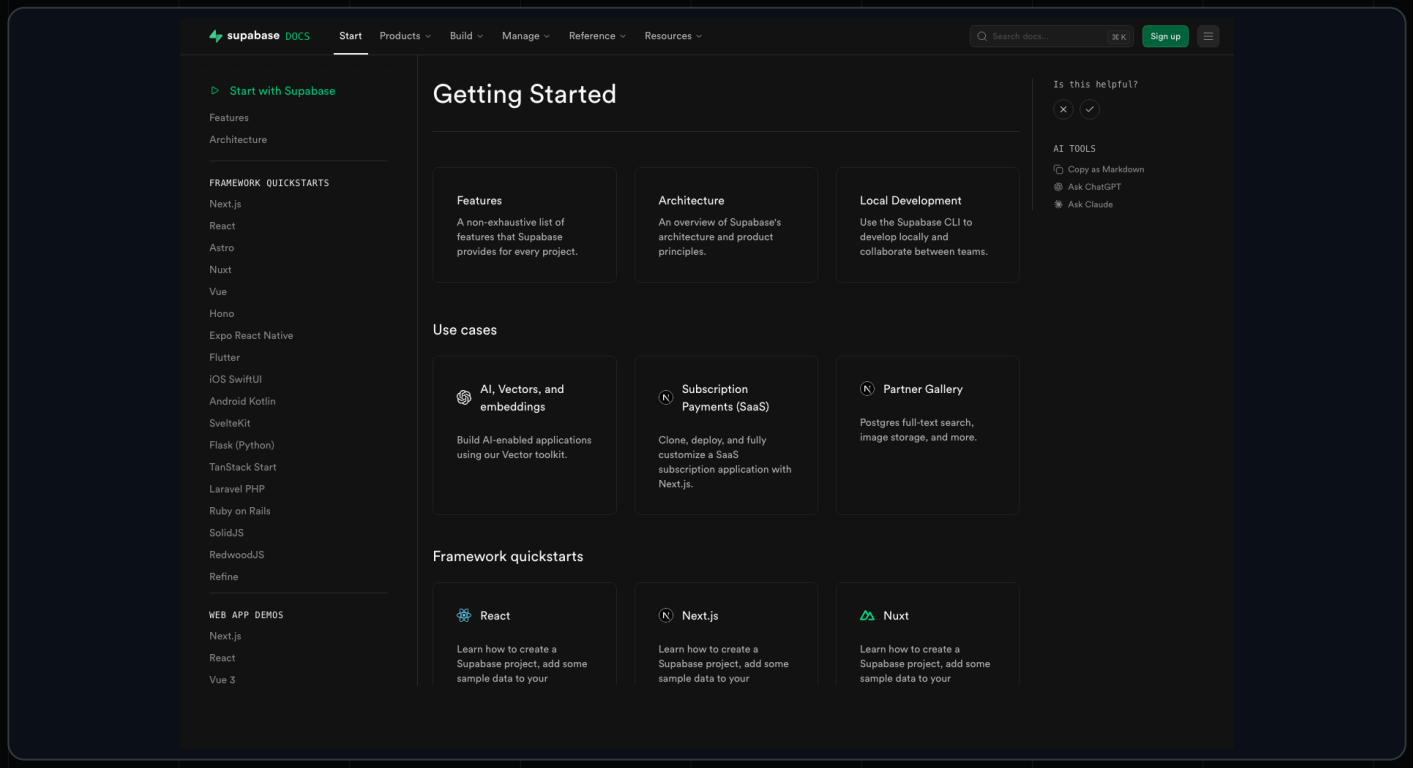
WHERE TO CLICK

Create a repo, push the project, use branches or commits for every meaningful change.

ENV NAMES

```
GITHUB_TOKEN only if an automation needs API access.
```

Supabase: click path for this route.



WHERE TO CLICK

Open app.supabase.com, click New project, choose org, name it, save the project URL and anon/publishable key.

ENV NAMES

`NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY, SUPABASE_SERVICE_ROLE_KEY`

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

INSPECT PROMPT

```
Inspect this repo for the Daily Operator Setup 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 Daily Operator Setup.
Outcome: Have a clean workspace that can safely run your first real automation every morning.
Tools allowed: Skool, GitHub, Supabase, Vercel, Resend, Telegram, PostHog, 1Password
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: Daily Operator Setup
Outcome: Have a clean workspace that can safely run your first real automation every morning.
Tools: Skool, GitHub, Supabase, Vercel, Resend, Telegram, PostHog, 1Password

# .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 Terminal. Type cd ~/Desktop && mkdir operator && cd operator and press Enter.

02 Type npx create-next-app@latest and press Enter. Press Enter again to accept the defaults: TypeScript yes, Tailwind yes, App Router yes, src directory yes.

03 When it finishes, type cd <folder-name> and press Enter. The folder name matches whatever you named the app.

04 Open chrome and go to supabase.com. Click Sign in, choose Continue with GitHub, then click the green New project button.

05 Name the project operator. Click Generate a password and save the password. Pick the closest region. Click Create new project. Wait about 2 minutes.

06 Click Project Settings in the left sidebar, then API. Copy the Project URL and the anon public key into a sticky note for the next step.

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

01 Go back to your editor. Create a new file in the project root called .env.local. Paste these two lines, replacing the values: NEXT_PUBLIC_SUPABASE_URL=your-url and NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key.

02 Open chrome and go to vercel.com. Sign in with GitHub. Click Add New Project. Import your operator repo from GitHub. If the repo is not listed, push your code to GitHub first.

03 During Vercel import, expand the Environment Variables section. Paste the same two variables from .env.local.

04 Click Deploy. Wait 1 to 2 minutes. Vercel gives you a live URL when the deploy turns green.

05 Back in Terminal in your project folder, type npm run dev and press Enter. Open chrome to http://localhost:3000 and confirm the starter page loads.

06 Take a screenshot of your live Vercel URL with the page rendering. Post it in the Beginner 04 Skool thread.

The build is not finished until verification is boring.

01 Can the routine run twice without manual fixes?

02 Can you revoke one key without losing the whole project?

03 Can you see failures from your phone?

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 Creating keys in too many places without a tracker.

02 Building without a run log, then not knowing what failed.

03 Confusing local success with production readiness.

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

Ship a daily brief that writes one row to Supabase and sends one message to either email or Telegram.

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

Skool: https://help.skool.com/

GitHub: https://docs.github.com/en/get-started

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

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

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