

Useful next: what we didn't cover

Things worth knowing as your apps grow. Each item: the name, then what it is or why it helps.

Getting more out of the agent

Audit before you change. Before a big change, ask: "Inspect this project, explain how it works, list the five biggest risks, and don't change anything." You learn the code and avoid breaking what works.

Plan mode. Press Shift+Tab in Claude Code to switch to plan mode: it researches and writes a plan without editing files. Approve the plan, then let it build.

Commit small, commit often. Save every working step in Git with a short message. If the next step goes wrong, you can go back. Claude Code can also undo its own recent edits with `/rewind`.

Parallel work with worktrees. A Git worktree is a second copy of your project on another branch. Two agent sessions can then work at the same time without overwriting each other.

Hooks. Small scripts Claude Code runs automatically at set moments, for example formatting code after every edit or blocking access to files with your keys.

Watch your usage. `/usage` shows how much of your plan is left, `/model` switches models. Use the cheaper model for routine work and save the strongest one for hard problems.

Making your app smarter

Structured outputs. When your app asks an AI for data, ask for JSON in a fixed format and check it before using it. Much more reliable than parsing free text.

Model routing. Use a cheap, fast model for simple jobs (labels, short summaries) and a strong one only where quality matters. OpenRouter makes switching a one-line change.

RAG (answers from your own documents). Split your documents into small pieces, find the pieces that match a question, and give only those to the model together with the question. Show the sources. Supabase can store this with pgvector.

Embeddings. Number "fingerprints" of text, used to find things with a similar meaning rather than the exact same words. The search step of RAG.

Evals. A small list of test questions with good answers. Re-run it whenever you change a prompt or model, so you notice when quality drops.

Prompt injection. Web pages, emails or user uploads can hide instructions that trick the AI. Never let the AI take real actions (send, pay, delete) based on such text without a check.

Background jobs. Long tasks (transcribing, big summaries) should run in a queue while the user gets a notification when done. Web requests on free hosting time out.

Webhooks. How other services tell your app something happened, like "payment succeeded". Check the signature and handle the same event arriving twice.

Automation tools. n8n (open source), Zapier or Make connect apps with visual flows, e.g. new sign-up, then add to a sheet, then send a welcome email.

Running a real product

Error tracking. Sentry (free tier) or your own error table with email alerts tells you about crashes before users do.

Uptime monitoring. Free services such as UptimeRobot check your site every few minutes and alert you when it is down.

Privacy-friendly analytics. See where users get stuck. PostHog has a free tier with EU hosting; Plausible and Umami work without cookies. Cookie-based tracking needs consent in the EU.

Rate limiting. Limit how often one user can call expensive features (for example 10 AI requests a minute) so a bug or a bad actor can't run up your bill. Upstash has a free tier.

Preview deployments. Vercel gives every pull request its own preview address. Test there before merging to the live site.

Rollback and feature flags. Vercel can switch back to the previous deployment in one click. Hide unfinished features behind an on/off switch instead of half-shipping them.

Database changes and backups. Change the database through numbered migration files, try them on a copy first, and know where your backups are. Free plans have limited backups; export before risky changes.

Accessibility. Readable contrast, keyboard use, labels for screen readers, large touch targets. Good for everyone, and the EU accessibility law (since June 2025) covers many consumer apps; very small businesses are exempt for services.

Languages. Keep all interface text in one place from day one, so translating later is easy. Never translate your product name.

Getting found. Page titles, a sitemap, link-preview images, and Google Search Console to see how people find you.

Money, law and free-tier traps

Charging money on Vercel. The free Hobby plan is for non-commercial use only. Move to Pro (\$20/month) before you take payments.

Merchant of record. Paddle, Lemon Squeezy, Polar or Stripe Managed Payments sell on your behalf and handle sales tax and VAT, for roughly 3.5-5% extra per sale. Much less paperwork for a first launch.

Registering and VAT. Selling usually means registering as a business in your country. In the EU, cross-border sales to consumers above 10,000 euros a year need VAT handled through the One-Stop Shop. Check with an accountant.

Privacy policy and GDPR. Say what data you collect, why, which services process it (Supabase, Vercel, OpenAI and so on) and how to delete an account. Add a working "delete my account" button.

Cancelling and refunds. EU consumers usually have 14 days to withdraw from an online subscription. Say clearly in your terms how cancelling and refunds work.

App store rules. Apple can reject apps that are "just a website in a frame", so add real phone features. Selling digital subscriptions inside iOS apps has its own payment rules.

Free-tier traps. Supabase pauses free projects after 7 idle days. Free private GitHub repos can't use branch protection or auto-merge. Vercel's free cron jobs run once a day. OpenRouter's free models allow 50 requests a day until you top up.

The bigger picture

MCP and A2A. MCP connects an agent to tools and data. A2A is a newer standard for separate agents to talk to and hand work to each other.

Multi-agent frameworks. LangGraph or CrewAI coordinate planner, coder and reviewer agents in your own code. Most solo projects don't need them yet.

Local models. Ollama or LM Studio run open models on your own computer: private and free per use, but weaker and they need a strong machine.

Claude Agent SDK. Build your own agents on the same engine as Claude Code, for example a support agent inside your app.