Deep Dive — everything explained

This is the long-form companion to `QUICK_START.md`. The Quick Start gets you live in two hours; this document explains *why* each piece exists and what to do when something behaves unexpectedly. Use it when something breaks weirdly, or when you want to understand what you're looking at.

Table of contents

Architecture at a glance

The kit is a single **Next.js 16 application** that runs on Vercel. It talks to:

- **Supabase** — Postgres database with Row-Level Security (RLS) + Auth + Storage (file uploads)
- **Stripe** — payments, hosted checkout, webhooks
- **Resend** — transactional + marketing email
- **GitHub** (optional) — collaborator invites for code-product fulfillment
- **Shippo** (optional) — shipping rates + label printing for physical products
- **Cloudflare Turnstile** (optional) — anti-bot CAPTCHA on auth forms

There are no other services in the critical path. The kit is intentionally a small set of well-known building blocks. The whole thing is one codebase, one repo, one deploy target.

The site is largely **server-rendered** (Next.js App Router with React Server Components). Client-side JS is small. Tailwind is used for styling. TypeScript everywhere.

The database (Supabase)

What's in the schema

44 historical migrations consolidated into `supabase/schema/current.sql`. Key tables:

Table	What it stores
`products`, `product_variants`, `product_images`, `categories`	catalog
`orders`, `order_items`, `order_status_history`, `payments`	purchases
`coupons`, `coupon_redemptions`	discount codes
`reviews`, `review_helpful_votes`	product reviews
`profiles`	user metadata + role (links to `auth.users`)
`webauthn_credentials`, `webauthn_challenges`	passkey registrations
`saved_addresses`	buyer address book
`refund_requests`	post-delivery refund queue
`processed_stripe_events`	webhook idempotency
`cart_sessions`	abandoned-cart recovery state
`email_subscribers`	newsletter signups
`referral_codes`, `referral_clicks`	affiliate program
`audit_log`	admin action audit trail

Storage buckets (auto-created by the schema):

- `product-images` — public, holds product preview images
- `product-files` — private, for digital downloads with signed 5-minute URLs
- `brand-assets` — public, for logo/favicon

Row-Level Security (RLS) — what it is and why it matters

Every table has explicit policies that say "user A can read their own orders, but not user B's". Even if a bug in the application code asks the database for someone else's data, the database itself refuses. It's defense-in-depth: the database is the last line of defense, not the first.

When you create a new table, **always enable RLS** and add policies before deploying. The kit's existing tables follow this pattern.

How to update the schema after launch

The kit ships 44 numbered migrations in `supabase/migrations/001_*.sql` through `044_*.sql`. The single drop-in file `current.sql` represents the *final state* of all of them.

When you make schema changes:

1. Write a new migration `045_my_change.sql` and apply it to your database
2. Mirror the change into `current.sql` so future fresh installs include it

If you ever want to drop your database and start over, you just paste `current.sql` into the SQL Editor again — it's idempotent (uses `CREATE IF NOT EXISTS` and `DROP CONSTRAINT IF EXISTS` everywhere).

Authentication and admin role

Sign-in flow

Default: email + password via Supabase Auth. Buyers register at `/login` (toggle to "Sign up"), confirm via email, then sign in. Sessions are JWT-based, stored in HTTP-only cookies (no XSS risk).

Custom Access Token Hook (one-time Supabase setup)

The kit uses a Postgres function `add_admin_claim` (created by `current.sql`) to inject an `is_admin` claim into the JWT. This makes admin checks fast — no database query per request.

To enable: Supabase → Authentication → Auth Hooks → **Custom Access Token** → Schema `public`, Function `add_admin_claim` → Save.

If you skip this, admin checks fall back to a database query — works but slower.

Promoting your first admin

Brand-new install has zero admins. After signing up as a normal user, run this SQL once:

```
UPDATE profiles
SET role = 'admin'
```

```
WHERE id = (SELECT id FROM auth.users WHERE email = 'YOU@example.com');
```

Sign out, sign back in, the new role takes effect. Future admins can be promoted from `/admin/customers` in the admin panel.

Multi-factor auth (TOTP + passkey)

The kit ships TWO independent second factors:

TOTP (authenticator app)

Standard 6-digit code from Google Authenticator, Authy, 1Password, etc. Powered by Supabase's native MFA.

Enable in Supabase: Authentication → Multi-Factor → toggle TOTP on → Save.

Enroll yourself: `/account/security` → "Add authenticator app" → scan QR → enter 6-digit code.

Enforce for admin: set `ENFORCE_ADMIN_MFA=true` in Vercel env. Admin pages will refuse to load until you complete a TOTP challenge in your session. **Enroll first, then flip the flag.** Otherwise you lock yourself out.

Passkey (Touch ID / Face ID / hardware key)

Newer, phishing-resistant. Powered by `@simplewebauthn/server` + a custom step-up route, since Supabase doesn't ship native WebAuthn yet.

No Supabase config needed. Just visit `/account/security` → scroll to "Passkeys" → "Add passkey" → use Touch ID / Face ID / your hardware key.

When you click "Verify passkey", a 30-minute cookie marker (`passkey_verified`) is set. The admin gate accepts that cookie as MFA-equivalent — so passkey OR TOTP both satisfy `ENFORCE_ADMIN_MFA` .

Counter-rollback detection refuses signatures whose counter didn't advance — defense against credential cloning.

Payments (Stripe)

Required env vars

```
STRIPE_SECRET_KEY=sk_test_… or sk_live_…
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_… or pk_live_…
STRIPE_WEBHOOK_SECRET=whsec_…
```

The webhook

Path: `https://<YOUR-DOMAIN>/api/webhooks/stripe`

Events the kit handles:

Event	What happens
`checkout.session.completed`	Marks order paid, processes GitHub invites, sends order confirmation, captures payment intent ID
`charge.refunded`	Marks order refunded, sends refund email
`charge.dispute.created`	Records the dispute on the order, sets `dispute_status`
`payment_intent.payment_failed`	Marks payment failed, releases inventory reservations
`checkout.session.expired`	Releases reservations after a buyer abandons checkout

Idempotency: every webhook checks `processed_stripe_events` first. If Stripe redelivers the same event (which happens after any 5xx), the second delivery short-circuits with 200 — no double refunds, no duplicate emails.

Local webhook testing

Install the Stripe CLI: https://stripe.com/docs/stripe-cli

```
stripe listen --forward-to localhost:3001/api/webhooks/stripe
```

That command prints a `whsec_...` secret you can use as `STRIPE_WEBHOOK_SECRET` in `.env.local` for local development. Production uses a different secret (the one Stripe Dashboard shows after you register the endpoint).

Apple Pay / Google Pay

Both auto-appear at Stripe Hosted Checkout when:

- Card is enabled (default)
- Apple Pay / Google Pay are activated in Stripe Dashboard → Settings → Payment Methods
- Your domain is verified for Apple Pay (Settings → Apple Pay → "Add new domain")

The kit serves the Apple Pay verification file from `/.well-known/apple-developer-merchantid-domain-association` via a route — paste the file contents into admin → Settings → Theme → "Apple Pay verification" field. **No need to host the file separately.**

Buy Now Pay Later (Klarna, Affirm, Afterpay)

1. Enable in Stripe Dashboard → Settings → Payment Methods (one-click each)
2. Set `NEXT_PUBLIC_BNPL_ENABLED=true` in env

The kit then renders an "or 4 interest-free payments of $X" badge on PDPs for products ≥ $35. The actual BNPL methods come from Stripe Hosted Checkout, not the kit.

Stripe Tax (automatic US sales tax)

Set `STRIPE_AUTOMATIC_TAX=true`. Then in Stripe Dashboard → Settings → Tax → register for the states where you have nexus (most sellers don't have nexus in any state until they hit thresholds). Hosted Checkout will collect a billing address and add the right tax automatically.

If you skip this, you're responsible for sales tax — which is fine if you're under nexus thresholds in every state (most one-person shops are).

Order fulfillment

The kit supports four fulfillment methods, configured per-product:

Method	What it does	Setup
`download`	Buyer sees a "Download" button on the order confirmation page that goes to a public URL	Set `fulfillment_target` to a URL on the product
`github_invite`	Buyer is added as a read-only collaborator to a private GitHub repo	Need `GITHUB_TOKEN` env var; set `fulfillment_target` to `owner/repo-name` on the product
`notion_share`	Buyer sees a Notion share link on the confirmation page	Set `fulfillment_target` to a Notion share URL on the product
`storage_download`	Buyer gets a signed URL (5 min expiry) to a file uploaded to Supabase Storage	Upload via the admin product form's File field

GitHub invites — the most common path

Token requirements (Fine-grained PAT at https://github.com/settings/personal-access-tokens):

- **Repository access:** the repo(s) buyers will be invited to
- **Permissions → Administration:** Read and write
- **Permissions → Metadata:** Read-only (auto-added)

Buyers receive `permission: "pull"` — read/clone access only. They cannot push, modify, delete branches, or change anything in your repo. This is enforced server-side at `src/lib/github/server.ts:85`.

Storage downloads

For products where you upload a ZIP / file directly:

1. Admin → Products → New (or edit existing) → Fulfillment Method: "Storage download"
2. Upload the file in the same form
3. Buyers get a 5-minute signed URL on the order confirmation page; buyers in their account `/account/library` can re-request the URL

If you set `DOWNLOAD_LINK_SECRET`, guest (non-logged-in) downloads get an HMAC-signed 7-day link. Without it, guests fall back to email-match + time-window checks.

Email (Resend)

The kit sends:

Email	Trigger	Required env vars
Order confirmation	Successful payment via webhook	`RESEND_API_KEY`
Refund notice	Refund issued	`RESEND_API_KEY`
Welcome	First sign-up	`RESEND_API_KEY`
Shipping notification	Mark-shipped event	`RESEND_API_KEY`
Delivery notification	Shippo tracking webhook fires "delivered"	`RESEND_API_KEY` + `SHIPPO_WEBHOOK_SECRET`
Cart-abandonment recovery	Daily cron, for carts >24h old	`RESEND_API_KEY + CRON_SECRET + CART_RECOVERY_SECRET`
Refund-request received / approved / denied	Buyer files refund request via `/account/orders/[id]`	`RESEND_API_KEY` + `ADMIN_ALERT_EMAIL`

Admin fulfillment-failed alert	GitHub invite fails after retries	`RESEND_API_KEY` + `ADMIN_ALERT_EMAIL`

If `RESEND_API_KEY` is missing, all email is silently dropped (the rest of the app still works). For your own domain (so emails come from `you@yourdomain.com` not Resend's default), add the domain in Resend dashboard → verify the DNS records they show.

For deliverability, set up SPF + DKIM + DMARC at your DNS provider. Resend's Domains page tells you the exact records.

Physical shipping (Shippo)

Optional. Skip if you sell only digital products.

When set up, the kit:

- Quotes shipping rates at checkout via Shippo's Live Rates API
- Buys the cheapest reasonable label when admin clicks "Mark shipped"
- Sends tracking-status emails to the buyer as Shippo tracking webhooks fire

Setup:

1. Sign up at https://goshippo.com (free tier covers 30 paid labels/month)
2. Get API key from https://apps.goshippo.com/settings/api
3. Set env vars: `SHIPPO_API_KEY`, `SHIPPING_FROM_NAME`, `SHIPPING_FROM_LINE1`, `SHIPPING_FROM_CITY`, `SHIPPING_FROM_STATE`, `SHIPPING_FROM_ZIP`, optional `SHIPPING_FROM_PHONE`
4. (Recommended) Set up tracking webhook: Shippo → Webhooks → URL `https://YOUR-DOMAIN/api/webhooks/shippo` → copy the signing secret to `SHIPPO_WEBHOOK_SECRET`

For each physical product, set `is_physical=true` and the dimensions (length/width/height/weight) in the admin product form.

Cron jobs

The kit ships four scheduled jobs in `vercel.json`:

Path	Schedule (UTC)	What it does

`/api/cron/low-stock-check`	`0 9 * * *` (daily 9 AM)	Emails admin if any variant inventory drops below the configured threshold
`/api/cron/abandoned-cart`	`0 11 * * *` (daily 11 AM)	Sends recovery emails to buyers whose cart sat >24h with no checkout
`/api/cron/release-stale-reservations`	`0 3 * * *` (daily 3 AM)	Releases inventory reservations from expired Stripe sessions
`/api/cron/cleanup-stripe-coupons`	`0 4 * * *` (daily 4 AM)	Deletes one-shot Stripe Coupons (`temp-` prefix) older than 7 days

All four authenticate via `Authorization: Bearer ${CRON_SECRET}`. Generate a secret with `openssl rand -hex 32` and set it on Vercel.

If you don't set `CRON_SECRET`, the cron routes refuse with 403 — non-breaking for the app but you lose the scheduled jobs.

The admin panel

`/admin` (admin-role-only). Pages:

- **Dashboard** — onboarding checklist, recent orders, key metrics
- **Products** — CRUD, image uploads, variants, AI description writer (if `ANTHROPIC_API_KEY` set)
- **Categories**, **Collections** — taxonomy
- **Orders** — list + detail with refund, mark-shipped, retry-fulfillment, status history
- **Payments** — Stripe payment list with refund actions
- **Coupons** — discount code management
- **Reviews** — moderation queue (pending → approved/rejected)
- **Marketing** — email-subscribers list, newsletter export to Resend
- **Inbox** — contact form submissions
- **Referrals** — affiliate codes + payout reports
- **Blog** — posts CRUD with markdown editor
- **Customers** — user list, role promotion, ban
- **Analytics** — orders, revenue, conversion charts
- **Refund requests** — post-delivery buyer-initiated refund queue
- **Audit log** — every admin write with diff
- **Settings** — site copy, legal pages, theme, brand, sections, navigation, announcements

---

Theming, content, branding

All site copy lives in `site_config` (table). Edit at `/admin/settings/site-copy`. Changes are live on next page load — no rebuild needed.

Admin-uploadable: logo, favicon, brand colors (HSL hex). Covers the entire storefront.

The kit ships with two themes ("Bone & Moss" and "Sun & Shadow"). Theme A is default; B is gated behind `NEXT_PUBLIC_AB_THEME_ENABLED=true` for 50/50 A/B testing.

Security model

A non-exhaustive list of what the kit does to keep you and your buyers safe:

- **Strict CSP with nonce-based script-src + 'strict-dynamic'** — XSS protection
- **CSRF protection** via `isSameOrigin()` on every mutation route
- **Rate limiting** on auth, contact, search, address, refund, and cart APIs
- **Webhook signature verification** for Stripe, Shippo, and PayPal-style replay protection on Shippo (24h window)
- **HMAC-signed download URLs** for guest downloads (when `DOWNLOAD_LINK_SECRET` set)
- **Constant-time comparisons** for credential checks
- **Counter-rollback detection** for passkey (defense against cloned credentials)
- **Idempotent webhook processing** via `processed_stripe_events`
- **Admin audit logging** for all writes (refund, cancel, role change, data export, etc.)
- **Defense-in-depth via RLS** — even if app code is wrong, the database refuses to leak data
- **MFA enforcement** for admin (TOTP or passkey) when `ENFORCE_ADMIN_MFA=true`
- **PII redaction** in error logs (emails, card numbers, IP addresses redacted before they hit console / Sentry)
- **`security.txt` + public security policy** at `/.well-known/security.txt` and `/security`

For the full audit baseline, see `SECURITY.md`.

Going live: production checklist

In rough order:

- [ ] Custom domain on Vercel + DNS pointing at Vercel
- [ ] All required env vars set in Vercel Production scope
- [ ] Stripe in live mode, live keys in Vercel

- [ ] Stripe webhook configured with live endpoint URL
- [ ] Apple Pay domain registered (if you want Apple Pay)
- [ ] Resend domain verified (for branded send-from address)
- [ ] DMARC `p=quarantine` (or stricter) at your DNS — protects your domain reputation
- [ ] Cloudflare Turnstile site key + secret (anti-bot CAPTCHA on login)
- [ ] Sentry DSN (error monitoring) — free tier covers most one-person shops
- [ ] First admin promoted via SQL
- [ ] TOTP enrolled, passkey enrolled, both tested
- [ ] `ENFORCE_ADMIN_MFA=true` flipped (after enrollment)
- [ ] First product created and a $0.49 test purchase completed end-to-end (with 99% off coupon)
- [ ] CRON_SECRET set if you want background jobs
- [ ] Refund + cancel + reorder flows tested
- [ ] Privacy policy + terms updated to reflect your business
- [ ] Backup contact information set in `ADMIN_ALERT_EMAIL`

When all of these are done, you're production-ready.

For the operational checklist before each release, see `STARTER_CLEANUP.md`. For symptom → cause → fix recipes, see `TROUBLESHOOTING.md`. For one-feature-at-a-time integration walkthroughs, see `OPTIONAL_FEATURES.md`.