ALLIED BizTech SOLUTIONS

FIELD GUIDE · 2026

Rebuild the D2C stack yourself.
The plan we follow.

For Shopify Plus and D2C operators at $25M to $500M GMV who want to replace Klaviyo, Cloudinary, Algolia and Yotpo with one app on their own accounts: one build, one contract, one ship date.

→ Is it worth it, at your scale
→ One foundation, four modules
→ The architecture, in three layers
→ Migration and go-live checklists
→ One data model for four modules
→ A ten-week build plan
→ Four cutovers, in the right order
→ Where teams get stuck

01 · IS IT WORTH IT

Four tools, four meters. *One build replaces them.*

Klaviyo bills per contact, Cloudinary per credit, Algolia per record and per search, Yotpo per app. Each meter climbs on its own as you grow. Underneath sit AWS SES, S3 with a CDN, a search engine and Postgres. A bundled rebuild pays those directly and replaces four bills with one fixed build price.

SCALE	THE FOUR A YEAR	REBUILT: YEAR ONE / AFTER	SAVED
$25M GMV · 100K contacts	~$48,000	~$57,400 / $10,000	~$66K over 3 years
$80M GMV · 500K contacts	~$112,000	~$63,900 / $16,500	~$48K in year one
$250M GMV · 1.5M contacts	~$280,000	~$110,500 / $33,000	~$169K in year one

The page's three D2C profiles, from public rate cards. "Rebuilt" = direct billing on your own accounts, hosting and maintenance, and our fixed bundle price: $47,400 below 1M contacts, $77,500 (the top of the $62,000–$77,500 scale tier) from 1M. If your own team builds it, put your engineering cost in that place.

THE RULE OF THUMB

On a $48K stack, the bundle pays back in about 15 months and returns ~$66K over three years. **At $112K a year, it pays back in about six months, saves ~$48K in year one and ~$95K a year after: ~$239K over three years.** At 1.5M contacts, year one alone saves ~$169K.

Who you need

- **Senior engineers** who between them know Postgres, a job queue, the Shopify API, AWS SES and DNS, S3 behind a CDN, and a search engine. Our team ships this in 8–10 weeks; a team working between other priorities should plan for longer.
- **Your ecommerce or marketing lead**, every week: the segments, flows, search terms and review rules that matter, the pilot brand or category, each cutover call.
- **An owner after launch**: hosting, deliverability, the search index, the moderation queue and on-call, across all four modules.

One foundation, four modules. *Build the shared part once.*

Four separate rebuilds mean four apps, four dashboards and the Shopify integration built four times. One build shares all of that: $47,400 against $54,700 for four, in 8–10 weeks instead of ~16.

BUILD ONCE, SHARED BY ALL FOUR

- **Shopify webhooks, ingested once**: orders, customers, carts, browse
- **One customer and order store** every module reads
- **Shared sign-in**: SSO, role-based permissions per module
- **Unified analytics**: one revenue-attribution layer
- **One ops dashboard**, one deploy pipeline

REBUILD ONE FOR ONE

- **Email** (Klaviyo): segments, automation flows, templates, campaigns, suppression, on SES and Postmark
- **Assets** (Cloudinary): an asset library, on-the-fly transforms in Cloudinary's URL grammar, on S3 with CloudFront or Cloudflare
- **Search** (Algolia): autocomplete, facets, ranking, on self-hosted Meilisearch
- **Reviews and loyalty** (Yotpo): review collection and widgets, points, tiers, referrals

REBUILD, SIMPLER

- **Ranking A/B tests**: a harness on your own index
- **SMS**: Twilio direct, replacing Postscript or Attentive (a priced add-on)
- **Several brands**: isolated workspaces in one app (a priced add-on)

DROP ON PURPOSE

- **Each vendor's long tail**: transforms your URLs never use, multivariate tests
- **Proprietary ML**: Algolia Recommend, Klaviyo's predictive CLV, Cloudinary's smart crop
- **Yotpo's cross-merchant network**: it does not move with you
- **Four logins and four renewal dates**

ADD WHAT FOUR VENDORS CANNOT

- Revenue attribution across all four modules
- One customer view: email, search, reviews, points

03 · ARCHITECTURE

Three layers. *One app on your accounts.*

Each tool is a UI on a commodity part: Klaviyo on SES, Cloudinary on S3 and a CDN, Algolia on a search index, Yotpo on Postgres and a webhook engine. Rebuilt, the four share one database, one Shopify stream and one sign-in, on your AWS and Cloudflare.

LAYER 1 · SHOPPERS SEE

The storefront

- Search box, autocomplete, facets
- Images from your own domain
- Review widgets and stars
- Loyalty and referral pages
- Flows, campaigns, receipts

LAYER 2 · YOUR TEAM USES

The ops app

- Email: segments, flows, templates
- Assets: library and transforms
- Search: ranking, synonyms, rules
- Reviews: moderation; loyalty admin
- One analytics view, four modules
- SSO and roles across all four

LAYER 3 · YOU RUN

Direct billing

- **AWS SES**: ~$0.10 per 1,000 emails
- **Postmark**: transactional
- **S3 + CloudFront**: assets
- **Cloudflare Workers**: ~$0.50 per 1M requests
- **Meilisearch**: a $40-a-month VPS
- **Postgres** on RDS or Supabase
- **Twilio**: ~$0.0079 per SMS

Decisions to make in week one

DECISION	OUR DEFAULT	WHY
Where it runs	Your AWS and Cloudflare, in Docker images	The same images run on AWS, GCP, Azure or on-prem
Shopify data	One webhook receiver into a queue, plus a nightly reconcile	Built once, read by all four modules
Image URLs	Cloudinary's grammar on your own image domain	You swap a domain; the front-end does not change
Search front-end	Meilisearch behind the InstantSearch adapter	A config swap, not a rewrite
Sending identity	A dedicated SES IP, warmed 2–3 weeks; Postmark for receipts	Your reputation, on your account

Shopify Plus contracts do not dictate your email, asset, search or reviews tool. The app uses Shopify's public webhooks and API; your Plus contract is unaffected.

One customer, one order. *Four modules read them.*

Model the shared core first, then each module's tables on top. Shopify's IDs are the join keys; every vendor's ID rides along as a column, so each import can run twice and every row traces back to its source.

TABLE	KEY COLUMNS
`customers`	id, shopify_customer_id, email, phone, klaviyo_id, yotpo_id, properties (jsonb), orders_count, total_spent, last_order_at
`orders`, `products`	shopify_order_id, customer_id, lines (jsonb), total, placed_at, delivered_at; shopify_product_id, title, variants, in_stock
`customer_events`	customer_id, type (viewed, carted, searched, reviewed, redeemed), module, at
`webhook_log`	shopify_webhook_id, topic, payload, received_at, processed_at, status
`consents`, `suppressions`	customer_id, channel (email, SMS), source, opted_in_at, withdrawn_at; email, reason, at
Email: `segments`, `flows`, `sends`, `message_events`	rules (jsonb); steps, exit rule; provider_message_id, ip_pool; event type (delivered, click, bounce, complaint)
Assets: `assets`, `usage_daily`	public_id (the S3 key), version, bytes, width, height; url_prefix, requests, cache_hit_ratio
Search: `search_events`, `click_events`	query, filters, hits_count, latency_ms; document_id, position, converted
Reviews: `reviews`, `loyalty_ledger`, `referrals`	order_id, product_id, rating, verified, status; points (+/−), reason; code, referee, reward_issued_at
`attributions`	order_id, module, touch (send, search click, review, reward), rule, value

TWO RULES THAT SAVE A REBUILD

No module keeps its own copy of a customer or an order, or attribution across them never adds up. **Credit each order by one written rule**, stored on the order, so every report shows the same number.

Ten weeks, *pilot included.*

This is the order we build in: the foundation first, the four modules on top of it, then a two-week pilot on one brand or category and four cutovers a week apart. Every module runs beside its vendor for at least two weeks before it takes over.

WEEK	BUILD	DONE WHEN
1	**Foundation.** Postgres, SSO and roles, hosting, the Shopify webhook stream and backfill. Exports from all four vendors into staging. SPF, DKIM, DMARC and SES production access.	A Shopify order lands in Postgres in seconds; export counts match each vendor
2	**Search and assets.** The product index in Meilisearch behind the InstantSearch adapter; originals copied to S3 under their public_id; the transform Worker.	Top queries return Algolia's results; production image URLs render the same
3	**Reviews and loyalty.** Review requests, widgets and star markup; the points ledger, tiers, rewards, referrals; the Yotpo import.	Review counts, averages and point balances match Yotpo's
4	**Email.** Segments, flows with timing and exit rules, templates, campaigns, suppressions, A/B tests. The SES IP starts warming.	Each segment returns Klaviyo's population on a sample
5	**The shared layer.** Revenue attribution across modules, the ops dashboard, alerts. Search, assets and reviews start running beside their vendors.	Every attributed dollar traces to a Shopify order
6–7	**Pilot.** Two weeks on one brand or category, with a daily fix list. Email joins the parallel run on the warmed IP. Search cuts over at the end of week 7.	Pilot numbers hold against the vendors
8–10	**Cutovers, a week apart.** Assets in week 8, reviews and loyalty in week 9, email in week 10, once inbox placement matches Klaviyo's.	Each vendor read-only, then closed

GOING ONE TOOL AT A TIME

Many teams start with the largest line item, usually Klaviyo, and add the other modules later. Build the shared foundation in the first project anyway: **every later module then plugs into it** instead of bringing its own Shopify integration.

Four cutovers. *One customer record.*

Each module on its own is a known build. The hard part is moving four live systems off four vendors without losing an order, a review, a point or an inbox, while every module reads the same customers.

1. **Ingest Shopify once, and trust nothing twice.** One receiver checks each webhook's HMAC signature, stores the raw payload, drops duplicates by webhook ID and hands the work to a queue. Webhooks arrive late, twice or out of order: keep the newer `updated_at`, and reconcile against the Shopify API every night.
2. **Match every vendor row to one customer.** Shopify's customer ID first, then the normalized email. Klaviyo's and Yotpo's IDs become columns. A row that matches nobody goes to a review list, never into a new customer.
3. **One sender per customer.** While a module runs beside its vendor, only one system emails a given customer: split by brand, category or segment. Review requests go out from one system per order.
4. **Shadow the loyalty ledger.** Earn points from the order stream in parallel, compare every balance with Yotpo's each day, and import the final delta at cutover. Customers see the same balance before and after.

Lowest risk first

#	MODULE	WHY IN THIS PLACE	ROLLBACK
1	**Search** (Algolia)	No customer-facing change in the migration	Point the adapter back at Algolia
2	**Assets** (Cloudinary)	URL-compatible: no front-end change	Point the image domain back
3	**Reviews, loyalty** (Yotpo)	Visible to shoppers, easily rolled back	Swap the widget script back
4	**Email** (Klaviyo)	The most sensitive: every inbox	Flows paused in Klaviyo, not deleted

Each cutover gets a one-week pilot before the next starts. Email goes last: 2–3 weeks of warm-up on a new dedicated SES IP, 2–4 weeks in parallel with Klaviyo, and the switch only when inbox placement matches. If a module's pilot fails, that module stays on its vendor until it is ready.

Move it, launch it, *run it.*

Migration

- ☐ Backfill Shopify customers, orders and products first
- ☐ Export all four vendors in week one, the changes since at each cutover
- ☐ Klaviyo: profiles with consent, segments, flows, templates, suppressions
- ☐ Cloudinary: originals under their public_id, named transformations
- ☐ Algolia: index settings, synonyms, rules, your top queries
- ☐ Yotpo: reviews with dates and photos, point balances, referrals
- ☐ Match every vendor row to one customer
- ☐ Check counts per vendor, then read a sample end to end

Go-live, per module

- ☐ Two weeks beside the vendor, numbers side by side
- ☐ A written rollback, tested once
- ☐ Search: top queries green; assets: production URLs identical
- ☐ Reviews: counts and averages match per product
- ☐ Email: SPF, DKIM, DMARC passing; IP warmed; placement matched
- ☐ Alerts on webhook lag, queue depth, bounces and errors
- ☐ Backups proven by a restore

Running it

- ☐ **Always on:** webhook and queue alerts, security patches
- ☐ **Weekly:** revenue by module, flow and search term to the ecommerce lead
- ☐ **Monthly:** deliverability, zero-result queries, cache-hit ratio, the moderation queue
- ☐ **Quarterly:** direct bills against the four tools you replaced; open loyalty points
- ☐ **Yearly:** a restore drill, an access review, DKIM keys rotated

Who owns it

Name one person before launch. One app now carries your email revenue, your product search, your images and your reviews: someone has to answer when the Shopify stream stalls or a flow stops at 2 a.m.

Six ways a stack rebuild *stalls.*

1. Four projects instead of one

Run as four separate rebuilds, the same scope costs ~$54,700 and takes ~16 weeks with one team, and you end up with four apps, four dashboards and the Shopify integration built four times. One build costs $47,400 and ships in 8–10 weeks.

2. Four copies of the customer

When each module keeps its own customer table, revenue attribution across email, search, reviews and loyalty never adds up, and the one view the rebuild promised never arrives.

3. Moving the biggest bill first

Klaviyo is the largest line item, so it is tempting to move it first. Inside one build it moves last: it touches every customer's inbox. Search and assets go first, because shoppers never see the switch.

4. Cutting over cold

Switching a module without running it beside its vendor is how review counts reset, images break and inbox placement drops. Two weeks in parallel, then the switch.

5. Two systems emailing one customer

During the parallel run, a shopper can get the same abandoned-cart email twice, or two review requests for one order. Decide who sends to whom before the first send.

6. The cheap build

A freelance build gets you a sender, a bucket and an index. Deliverability, URL compatibility, relevance parity, review integrity and the shared attribution layer stay on you, and those are what the four vendors charge for.

And one that is not a mistake: keeping one tool

Love one of the four? Keep it and rebuild the other three; the bundle price drops by ~$10–14K. Yotpo is the one teams most often keep, for its review data and widgets.

RATHER HAVE IT BUILT?

We build this exact plan. *Fixed price.*

The same foundation, four modules and cutover order, delivered by a senior team that has shipped 250+ products.

$47,400 fixed for all four modules, $7,300 less than four separate rebuilds

Live in 8–10 weeks, after a two-week pilot on one brand or category

All four migrated: segments, assets, indexes and reviews

90-day warranty, four check-ins in year one, code in your GitHub

We can run it for $8,000–$12,000 a year, all four modules

1M+ contacts $62,000–$77,500; **3–6 brands** +$16,400–$25,500

alliedbiz.tech/rebuild/d2c-stack-rebuild

Allied BizTech · 15+ years · ISO 9001:2015 · 250+ products shipped. Klaviyo, Cloudinary, Algolia, Yotpo, Shopify, Shopify Plus, Postscript, Attentive, AWS, Amazon SES, Amazon S3, CloudFront, Cloudflare, Postmark, Twilio, Meilisearch, Supabase and InstantSearch are trademarks of their owners, named here for identification only; this guide is independent and not endorsed by any of them.