

FIELD NOTES / 01



The Lean Automation Stack for D2C Operators

A field guide to the operations layer — four layers, seven workflows, and the decisions worth automating.

In this issue

01 The four-layer stack

Storefront, runtime, customer surface, visibility — what belongs in each layer, and what to delete.

02 The first 7 workflows

The automations to ship first, each with a trigger, an outcome, and the guardrail that stops it doing harm.

03 Where AI earns its place

The five steps where AI reliably pays back, the ones where it doesn't, and how to pilot one safely.

04 The decision triggers

The thresholds worth encoding — hide a SKU, escalate a ticket, fire a win-back, alert on margin and stock.

05 The integrations register

How to track tokens, owners, and rotation dates so an expired auth never kills a launch week.

06 What to delete

The tools and steps removed in most operations builds — and what quietly replaces them.

A working guide, not a white paper. Read it with your own store open in the next tab.

How to use this

You run a direct-to-consumer store doing somewhere between \$100k and \$5M a year on Shopify or WooCommerce. You have products people want. What you may not have is a clear map of how orders, stock, support, and reporting actually move through the business.

The premise of this guide is simple: most stores at this stage are under-engineered, not under-marketed. Another ad account or a fifth email flow rarely fixes a revenue problem that is really an operations problem. Money leaks in the gaps between tools — the order that shipped late, the refund that sat for four days, the out-of-stock SKU still taking ad spend, the report nobody quite trusts.

What follows is the operations layer, written the way we build it: four layers to name your stack, seven workflows to ship first, the decisions worth automating, and an honest account of where AI helps and where it does not.

How to read it

- Skim the four-layer stack so you can name where work lives.
- Pick two workflows from the list of seven and ship them this month. Not all seven at once.
- Build the integrations register before your next launch — it is five minutes that saves a launch day.

Most stores are under-engineered, not under-marketed. The operations layer is where the money leaks.

One rule sits underneath all of it: every automation needs an off switch and a guardrail. Automation without a guardrail is just a faster way to make the same mistake a thousand times.

Four layers, named

Every store, whatever its size, runs on four layers. Name them and you can see where work actually lives — and where it is quietly duplicated. The goal for each layer is the same: do one job well, and stay replaceable without touching the others.

1 • Storefront

Where the customer buys — Shopify or WooCommerce, the theme, checkout, payment, and the catalog. This layer should be boring and fast. It is not the place for clever logic; every app that injects script into checkout is a tax on conversion. Keep: catalog, pricing, checkout, payment providers. Delete: page-builder bloat, duplicate analytics tags, and apps installed for one campaign and never removed.

2 • Runtime / automation

The layer that does work when something happens — n8n (self-hosted or cloud) or Make, sitting between your storefront, suppliers, help desk, and data. It listens for events and takes action. Keep: webhooks, scheduled jobs, and the logic that used to live in someone's head or a manual checklist. Delete: Zapier chains that fan out across ten single-step zaps — collapse them into one flow you can read end to end.

3 • Customer surface

Every place you talk to the customer after the sale — email and SMS (Klaviyo), the help desk (Gorgias, Zendesk, Help Scout), order-status pages, and reviews. Keep: transactional messages, lifecycle flows, support conversations, the review request. Delete: the second email tool you kept “for newsletters,” and the overlapping abandoned-cart flows that double-send.

4 • Visibility / reporting

The layer that tells you the truth — one dashboard pulling orders, margin, stock, refund rate, and support load into a single view you check daily. A lightweight BI tool (Metabase, Looker Studio) or a well-built sheet fed by the runtime layer. Keep: the ten numbers that drive decisions. Delete: the 40-tab spreadsheet nobody opens, and every “report” trapped inside one app that can't be compared to anything.

LAYER	ITS JOB	TYPICAL TOOLS	FIRST THING TO DELETE
Storefront	Sell, fast	Shopify, WooCommerce	Unused checkout apps
Runtime	Act on events	n8n, Make	Zapier spaghetti
Customer surface	Talk after the sale	Klaviyo, Gorgias	The second email tool
Visibility	Tell the truth	Metabase, Looker Studio	The 40-tab sheet

Each layer should be replaceable without touching the others. If swapping your help desk means rewriting your order logic, the seams have bled.

Read your own stack in five questions

Before you add anything, find out what you already have. Answer these honestly and the first fixes name themselves.

- **Storefront.** How many apps inject script into your checkout? If you can't list them from memory, that number is quietly costing you conversion.
- **Runtime.** What still runs only because a person remembers to do it at the same time every day? That task is your first workflow.
- **Customer surface.** How many tools can email or text your customer — and do any two of them fire on the same trigger?
- **Visibility.** Is there a number your team quietly disagrees on because two tools report it differently? That's a source-of-truth problem, not a math problem.
- **The seams.** If you swapped one tool tomorrow, how many others would break? The answer is how tangled the stack really is.

Ship these first

Ship in order of pain, not order of interest. Each workflow has three parts: the trigger that starts it, the outcome it produces, and the guardrail that stops it doing harm. If you cannot name the guardrail, the workflow is not ready.

1 • Order → fulfillment handoff

TRIGGER	<code>order.paid</code> (storefront webhook)
OUTCOME	The order is pushed to your 3PL or supplier in their format, tagged, and a fulfillment record is created — no copy-paste, no CSV at 6pm.
GUARDRAIL	Hold anything flagged for fraud review, above a high-value threshold, or with a mismatched billing/shipping country. Those route to a human queue, not straight to the supplier.

2 • Low-stock alert (and auto-hide)

TRIGGER	<code>inventory.updated</code> → <code>qty <= 5</code>
OUTCOME	A message to the ops channel with the SKU, days of cover left, and a draft reorder. Optionally set “continue selling = off” or hide the SKU at zero.
GUARDRAIL	Never auto-hide a SKU with unfulfilled paid orders, and never reorder automatically — a human confirms the PO. The flow surfaces the decision; it does not spend money.

3 • Refund / return triage

TRIGGER	<code>return.requested</code> <code>refund.created</code>
OUTCOME	The request is classified (wrong item, damaged, changed mind, not delivered), order history is attached, and small goodwill refunds under a set value auto-approve while the rest escalate.
GUARDRAIL	Any refund above a dollar threshold, or a second refund for the same customer inside 60 days, always goes to a human. Money out is never fully automated.

4 • Abandoned-checkout recovery

TRIGGER	<code>checkout.created</code> → no <code>order.paid</code> in 45–60 min
OUTCOME	A short, useful recovery sequence — one reminder, one with a reason to return, one final — with recovered revenue attributed so you know its worth.
GUARDRAIL	Suppress anyone who completed via another channel, opened a support ticket, or already got the sequence in the last 14 days. Never dun a customer who already paid.

5 • Post-purchase sequence

TRIGGER	<code>order.fulfilled</code> (first-time vs repeat)
OUTCOME	Shipping confirmation, a genuinely useful “how to use it” message, then a review request timed to arrive after the product does — not the day it ships.
GUARDRAIL	Gate the review request on delivery, not dispatch. Exclude orders with an open ticket or a refund in progress. Cap total messages so a three-item customer isn't hit three times.

6 • Support ticket triage / classification

TRIGGER	<code>ticket.created</code> (help desk)
OUTCOME	The ticket is tagged by intent (where-is-my-order, return, product question, wholesale), prioritized, and the customer's order data is attached so the agent opens a full picture, not a blank reply.
GUARDRAIL	Classification assists; it does not auto-reply on anything involving money, complaints, or safety. Low-confidence tags route to a human, not to a guessed macro.

7 • Weekly ops digest

TRIGGER	<code>schedule: every Monday 07:00</code>
OUTCOME	One message with the numbers that matter — revenue, orders, AOV, refund rate, support backlog, stockouts, and the three SKUs trending down — sent to the people who act on them.
GUARDRAIL	The digest reads from the reporting layer only. If a number can't be sourced and reconciled, it is marked “unverified” rather than shown as fact. A dashboard that lies quietly is worse than none.

Which two to pick first

Two of these, shipped and stable, will change how the week feels more than any new tool. Which two depends on where you are:

- Just crossed \$100k, still hands-on. Start with #1 (order → fulfillment) and #7 (weekly digest). Get the daily grind off manual, and start seeing numbers you've been running blind on.
- Scaling, ad spend on. Start with #4 (abandoned-checkout) and #2 (low-stock). Protect the revenue you're paying to acquire, and stop spending on things you can't ship.
- Plateaued or margin-squeezed. Start with #3 (refund triage) and #6 (support triage). Close the cost leaks and reclaim the hours support is quietly eating.

Whatever you pick, ship one fully before you start the second. A half-built workflow is a liability, not an asset.

AI is a component, not a strategy

In an e-commerce operation, AI pays back reliably in a handful of specific places — all of them tasks where “roughly right, reviewed” beats “slow and perfect,” and none of them touching money or compliance without a human in the loop.

Where it earns its place

- Support triage and classification. Sorting tickets by intent, sentiment, and priority, and drafting a first reply for an agent to approve. High volume, forgiving of the odd miss, human still sends. The highest-return use in most stores.
- Product data cleanup. Normalizing messy supplier feeds — titles, attributes, categories, missing dimensions, inconsistent casing. AI drafts, a rule validates, a human approves the batch.
- Review and message drafting. First drafts of review responses, canned-but-personal replies, and post-purchase copy. The model gets you 80 percent; the brand voice and the send stay human.
- Content variants at scale. Generating and testing subject lines, ad angles, and product-description variants. Cheap to produce, judged by results, low cost when a variant is weak.
- Anomaly flagging. Reading the daily numbers and flagging what looks off — a refund-rate spike, a SKU converting far below its class, a sudden channel drop. It is a smoke alarm, not a firefighter.

Where it doesn't

- Anything that moves money without review — issuing refunds, changing prices, releasing payouts, editing payment settings. The failure mode is expensive and irreversible.
- Compliance and legal text — tax, safety claims, ingredient or regulatory copy, terms. These must be exactly right, and “confident and wrong” is the model's worst habit.
- Exact inventory math — final reorder quantities, committing stock. Use AI to surface the signal; keep the arithmetic and the commit deterministic.
- Final sends on sensitive topics — complaints, disputes, anything involving a person's money or a bad experience. Draft, yes. Auto-send, no.
- The single source of truth — never let a model be the system of record. It drafts into a system; it is not the system.

Let AI draft, classify, and flag. Keep humans on approve, send, and anything with a dollar sign or a legal line.

How to pilot an AI step without getting burned

When a use case above fits, roll it out like an engineer, not a gambler. Five steps make the difference between leverage and a mess:

- Shadow mode first. Run it beside a human for two weeks — it predicts, the human still decides. Compare the two before it touches anything live.
- Set a confidence floor. Below it, route to a human. Start high (0.85) and lower it only as the track record earns your trust.
- Keep a human on the send for anything customer-facing until the error rate is boring. Boring is the goal, not perfect.
- Log every decision — input, output, and confidence. An AI step you can't audit is one you can't improve or defend.
- Build the kill switch before the feature. One toggle back to the manual path. If you can't turn it off in five seconds, it isn't ready to turn on.

Decisions worth encoding

Automation is just decisions someone wrote down. These are the ones worth encoding first, with example thresholds. Treat the numbers as starting points — tune them to your margins and volume, then leave them alone long enough to trust them.

DECISION	FIRE WHEN	ACTION	CONFIRM
Hide / flag a SKU	Stock = 0 with no inbound PO, or return rate > 15% over the last 30 units	Hide from collections; pause its ads	Ops, same day
Escalate a ticket	Classifier confidence < 0.7, or message mentions refund / chargeback / "broken" / legal, or 2nd contact on one order	Route to priority queue; skip auto-reply	Human agent
Fire a win-back	Days since last order > 2x the customer's usual gap (or 90 days for one-time buyers)	Enter win-back flow with a real reason to return	Auto; suppress if ticket open
Alert on margin	Blended margin < 15% after COGS, shipping, and fees — often a discount stack	Alert ops; flag the discount rule	Human before any change
Alert on stock cover	Days of cover < lead time + safety buffer (e.g. cover < 21d on a 14d lead)	Draft a reorder; notify ops	Human places the PO
Pause spend	SKU out of stock, or its 7-day margin turns negative	Pause the ad set / drop from catalog feed	Auto pause, human restart

Two guardrails apply to all of them

- Every threshold has a cooldown so it doesn't fire twenty times an hour.
- Every automated decision writes a log line — what fired, when, and what it did — so you can audit the machine's reasoning after the fact.

How to tune a threshold

Start deliberately loose, then tighten. Set the number, watch it for two weeks, and track one thing: how often it fired when it shouldn't have. If false alarms annoy the team, loosen it; if real problems slip through, tighten it. Once it has been quiet-but-right for a month, stop touching it. A threshold you re-tune every week isn't a rule — it's a distraction.

The launch-week insurance policy

The quiet killer of launch weeks is an expired token. A Klaviyo key rotated, an app's scope changed, a supplier password reset by someone who didn't know it was load-bearing — and a flow fails silently until someone notices revenue is down. The fix is boring and it works: a register.

It is a single table — a sheet, a Notion database, an Airtable base — listing every integration, who owns it, what it is for, when its credential expires, and when it was last checked. One person owns the register. It gets a five-minute review every month and a hard check before any launch.

INTEGRATION	OWNER	TOKEN / EXPIRY	PURPOSE	LAST CHECKED
Shopify Admin API	Ops lead	Vault • rotate 2026-10-01	Order + inventory sync	2026-07-14
Klaviyo	Lifecycle	Vault • no expiry	Email + SMS flows	2026-07-12
n8n → 3PL	Ops lead	Vault • rotate 2026-09-15	Fulfillment handoff	2026-07-15
Gorgias	Support	Vault • rotate 2026-08-30	Ticket triage	2026-07-10
Meta CAPI	Growth	Vault • rotate 2026-08-01	Server-side events	2026-07-11
Supplier feed	Ops	Vault • rotate 2026-09-01	Product + stock import	2026-07-13

Four rules that keep it honest

- Credentials live in a secrets manager, never in a workflow node, a shared doc, or a screenshot. The register points to the vault; it does not hold the secret.
- Every integration has a named human owner. “Marketing owns it” is how a token dies unnoticed.
- Add a rotation date the day you create a credential, not the day it expires.
- Before any launch or sale, run the register top to bottom. Five minutes against a lost launch day is the best trade in the building.

Most builds remove more than they add

Complexity is the cost of a stack; fewer moving parts doing more reliable work is the goal. These are the removals that come up in almost every operations build — what goes, and what replaces it.

- Hand-updated spreadsheets → one dashboard fed automatically by the runtime layer. If a human retypes a number that already exists in a system, that's a workflow waiting to be built.
- Zapier spaghetti — a dozen single-step zaps nobody can trace → one n8n or Make flow you can read top to bottom, with error handling in one place.
- Three half-used apps doing overlapping jobs → one native feature or one deliberate tool. Every app is a subscription, a login, a support surface, and a script on your store.
- Duplicate analytics and pixels — two GA tags, three pixels → one clean measurement setup, server-side where it matters. Double-counting is worse than not counting.
- Overlapping email flows that double-send → one flow with clear entry and exit rules.
- The “temporary” campaign app installed for one promo and never removed → gone. Uninstall is a monthly habit, not an annual cleanup.
- Checkout scripts you can't explain → if no one knows why a script is in checkout, it comes out. Conversion lives here; guard it.

A build is finished not when there is nothing left to add, but when there is nothing left to remove.

Before adding any tool, ask what it replaces. If the answer is “nothing, it's additive,” that is usually a reason to stop, not to buy.

Keep it alive

A stack is only as good as the habit around it. Four rhythms keep an automated operation honest — and none of them runs longer than half an hour.

RHYTHM	TIME	WHAT YOU DO
Daily	5 min	Clear the human-review queue — held orders, escalated tickets, refunds over threshold — and scan the ops channel for alerts.
Weekly	15 min	Read the digest. Action the three SKUs trending down, and pull the support backlog older than 48 hours back to zero.
Monthly	30 min	Run the integrations register. Review thresholds and their false-positive rates. Uninstall one tool you no longer use.
Pre-launch	30 min	Register top to bottom. Fire one live test order through fulfillment. Confirm stock buffers, and that every automated message still renders.

The build is the easy part. The cadence is what makes it pay.

Miss a day and nothing burns down — that's the point of guardrails. Miss a month and the register goes stale, thresholds drift, and the quiet system stops being quiet. The cadence is cheap insurance on everything you built.

Start with what leaks

If you turn one thing here into action: pick the two workflows from section 02 that map to your worst weekly pain, name their guardrails, and ship them. Then build the register from section 05. That is a month of work that pays for itself.

If you'd rather map it with someone

Start with a Store Systems Audit. We walk your four layers, find where money leaks, and hand you a prioritized build plan you can run yourself or with us. It is \$297, and it is the whole picture — not a sales call.

Free, if you want to start today

- Store Systems Scorecard — rate your four layers in ten minutes and see where the gaps are.
- Automation ROI Calculator — put a dollar figure on the manual work you're about to automate.

Everything is at digitalshijil.com.

Quiet systems. Precise operators.

Digital · Shijil · digitalshijil.com · Field Notes / 01 · 2026 · v1.0