■ MISHA AI

Launch an AI agent for ecommerce

A practical Claude Code workbook for turning store data, specialist skills, and approval gates into a useful operator.

THE PROMISE

Not a chatbot. A repeatable loop: observe → reason → propose → approve → act → verify.

Download the companion checklist at getmisha.co/agent-launch

Start with the job, not the model

An ecommerce agent is a small operating system around a decision. It should know what it is watching, what good looks like, what it is allowed to change, and how a human confirms the change. Claude Code is the build surface; your repo is the operating memory.

What you will have by the end

A versioned repo with a CLAUDE.md contract, a small set of ecommerce skills, read-only connectors, fixture tests, approval payloads, cost limits, and a pilot checklist. The goal is a working first slice - not a demo that claims to run the whole company.

The six-part loop

01	Observe	Pull the smallest reliable slice of orders, spend, margin, inventory, or customer behavior.
02	Reason	Compare the slice to a target or constraint. Show assumptions.
03	Propose	Return a ranked recommendation with upside, cost, confidence, and risks.
04	Approve	Require a human to approve any write, spend, message, or irreversible change.
05	Act	Execute an idempotent tool call with a dry-run and audit record.
06	Verify	Re-read the source of truth and report what changed.

Rule of thumb: if you cannot explain the approval boundary in one sentence, the agent is not ready for production.

1 / Set up the build surface

Claude Code reads a project, edits files, runs commands, and connects to development tools. Keep the agent in a dedicated repository so its instructions, skills, tests, and audit artifacts are versioned together.

Install and start

```
curl -fsSL https://claude.ai/install.sh | bash
mkdir ecommerce-agent && cd ecommerce-agent
git init
claude
```

Create CLAUDE.md

```
# Ecommerce operator
Goal: explain the next best action for our store.
Default mode: READ-ONLY. Never send, spend, publish, delete, or edit without approval.
Every recommendation: evidence, assumptions, impact, confidence, rollback.
Never print secrets. Never commit .env files.
Before any write: show a dry-run and ask for confirmation.
```

Create the first small contract

```
mkdir -p .claude/skills src tests/fixtures logs
touch src/metrics.ts src/report.ts
touch tests/fixtures/week.json

Ask Claude Code:
Read CLAUDE.md. Inspect the fixture. Propose a read-only
weekly performance report. Do not add network calls yet.
```

The first proof is local: a prompt, a fixture, a predictable report, and a test you can run without production credentials.

2 / Load ecommerce expertise

Finsi's awesome-ecommerce-skills repository is a curated, MIT-licensed collection of 178 ready-to-use skills across 17 categories. It covers marketing and growth, data and analytics, payments and checkout, Shopify, security, inventory, and more.

Install only what the decision needs

```
git clone https://github.com/finsilabs/awesome-ecommerce-skills.git
cp -r awesome-ecommerce-skills/skills/marketing-growth/cart-abandonment-recovery .claude/skills/
tessl install finsi/stripe-integration
```

Useful starter skills

- Checkout Flow Optimization - reduce friction before buying more traffic.
- Product Data Modeling - make variants, attributes, and metafields legible.
- Cart Abandonment Recovery - reason about timing and consent.
- Inventory Tracking - surface low-stock and multi-location risk.
- Customer Lifetime Value - prioritize retention by economics.
- E-commerce SEO - keep metadata and structured data consistent.

Examples: how to use a skill

```
With checkout-flow-optimization loaded:
Audit our Shopify checkout for the top three abandonment risks.
Use native settings first, cite the exact setting path, and return
a ranked test plan. Do not change the checkout.

With inventory-tracking loaded:
Find products with less than 21 days of cover. Show the formula,
lead-time assumption, and a draft reorder list. Do not place orders.
```

Do not load every skill by default. Start with the business decision, add the smallest relevant context, and re-check paths as the repository evolves.

3 / Build a read-only vertical slice

Start with one question: What changed in contribution margin this week, and what should we investigate next? The first agent reads data, computes a transparent comparison, and recommends a next move. It does not change an ad, product, price, or customer record.

Prompt the build

```
Read the repo and propose a read-only margin investigator.
Use installed ecommerce skills. Define input/output schemas,
missing-data behavior, three fixture tests, and the approval boundary.
Do not add a real API call until fixtures pass.
```

Stable output contract

```
{"question":"string","window":{"from":"date","to":"date"},"evidence":[{"metric":"string","value":"number
","source":"string"}],"recommendation":"string","confidence":"low|medium|high","needs_approval":false}
```

Test: happy path; missing COGS; conflicting Shopify and warehouse orders; promotion outlier.

4 / Connect tools with MCP

MCP gives Claude Code a controlled bridge to external tools. Treat every connector as a privilege boundary. Start with read-only resources, narrow scopes, and a sandbox account.

Sequence

- Name the source of truth for orders, spend, email revenue, and ledger data.
- Create one read-only tool per system; normalize dates, currency, and attribution.
- Keep write tools disabled until fixtures and review trails pass.
- Log request ID, actor, tool, inputs, result hash, approval, and verification.

Configuration sketch

```
claude mcp add --transport http shopify-readonly \
https://your-internal-gateway.example/mcp

# Ask Claude Code to list tools and document which are read-only.
```

Never paste production tokens into prompts. Store credentials in the provider or environment, restrict scopes, rotate them, and test revocation.

5 / Add actions only after approval

A recommendation is not authorization. Make the agent produce a plan first. A human approves the exact target, amount, audience, timing, and rollback. Then execute once and return a receipt.

Approval payload

```
ACTION PLAN
Target: Meta prospecting ad set
Change: reduce daily budget from $500 to $425
Why: 7-day blended MER below guardrail
Evidence: spend, revenue, margin, attribution
Risk: slower learning; rollback: restore $500
Reply APPROVE to execute or REVISE.
```

Guardrails

- Budget ceiling and daily spend delta limit.
- No customer message without copy preview and audience count.
- No delete; archive with a restore path.
- Idempotency key on every write.
- Kill switch disables all writes.
- Verify against the source of truth after the action.

6 / Test like an operator

The dangerous failure is a confident answer that is almost right. Test the reasoning boundary, not only the happy path.

Layer	Test	Gate
Unit	Calculations, currency, timezone, attribution	Fixtures pass
Contract	Tool schemas and error shapes	No silent fallback
Evaluation	Realistic merchant questions with expected evidence	Score stored
Safety	Prompt injection, secret leakage, retries	Blocked or escalated
Canary	One store, low-risk action, human on-call	Receipts reconcile

Have someone who did not build the agent try to break it. Give them a test store, not production.

7 / Common ecommerce tasks

Do not start with a general-purpose agent. Start with a task your team repeats every week. These prompts are deliberately narrow: ask for evidence, a proposed action, and a stopping rule.

Weekly performance brief

```
Compare the last 7 days with the prior 7 and the same period last year.
Explain changes in revenue, orders, AOV, contribution margin, spend, MER,
and returning-customer share. Cite source and timestamp for every metric.
Return the top 3 investigations; do not recommend a budget change yet.
```

Low-stock and cash-risk check

```
Find products with projected stockout inside lead time. Include units sold,
open purchase orders, days of cover, margin, and lost-revenue risk.
Flag missing lead times. Draft a reorder list; do not place an order.
```

Campaign QA before launch

```
Review this campaign against the approved brief. Check naming, tracking,
audience exclusions, landing URL, creative dimensions, claims, budget,
and frequency risk. Return PASS / NEEDS_REVIEW with exact fixes.
```

Lifecycle opportunity list

```
Segment customers by recency, frequency, monetary value, and subscription status.
Suggest one test for win-back, post-purchase, or replenishment. Include audience
size, suppression rules, consent requirements, expected upside, and stop rule.
```

Customer support triage

```
Classify these tickets by intent and urgency. Draft replies using approved policy.
Escalate refunds, chargebacks, safety issues, and angry VIP customers.
Never send a reply; return a queue with confidence and missing context.
```

8 / What it costs to run it yourself

The model bill is usually not the whole cost. Budget for the tool, the data plumbing, the reliability work, and the person who reviews decisions. Prices change, so treat these as planning ranges and re-check vendor pages before committing.

A realistic small-team budget

Cost bucket	Planning range	What drives it
Claude access	$20-$125 / seat / month	Team vs premium usage; Claude Code is included in paid plans.
API model usage	$10-$300+ / month	Tokens, context size, retries, model choice, and task frequency.
Hosting + jobs	$0-$100 / month	Serverless jobs, cron, queues, database, logs.
Connectors / vendors	$0-$500+ / month	Paid APIs, data warehouses, email/SMS, managed MCP gateways.
Human operations	4-20 hrs / month	QA, prompt updates, reconciliation, incident response, approvals.

A useful mental model: a 10,000-token input plus a 2,000-token output on a $3/$15-per-million-token model is about $0.06 before retries and connector costs. Cheap calls become expensive when you send the entire catalog every time.

Self-hosting pitfalls

• Token surprise: large context, repeated retries, and parallel jobs can multiply usage.

• Data surprise: warehouse and platform numbers disagree; reconciliation becomes your job.

• Permission surprise: a connector can expose more than the agent needs. Scope and rotate credentials.

• Reliability surprise: APIs rate-limit, time out, and change schemas. Build retries and alerts.

• Human-cost surprise: someone must review edge cases, update skills, and own incidents.

• Opportunity-cost surprise: a clever demo can delay the simple spreadsheet or dashboard that solves the problem.

Use a monthly hard cap, per-task budget, alerting at 50/80/100%, and a kill switch. The cheapest safe first version is read-only.

9 / Launch checklist

- One owner and one named business decision.
- CLAUDE.md defines read-only defaults, approval, and secret handling.
- Relevant skills are installed and reviewed.
- MCP tools are scoped and tested against fixtures or a sandbox.
- Outputs cite evidence, assumptions, confidence, and next check.
- Writes are dry-run first, idempotent, reversible, and verified.
- Logs contain no tokens or raw customer PII.
- Kill switch exists and has been tested.
- Pilot has a success metric, stop threshold, review date.
- A monthly cost cap, usage alert, and incident owner are written down.

30 / 60 / 90

0-30: read-only investigator, one source of truth, fixtures, evaluation set. 31-60: one low-risk action with approval and receipts. 61-90: connect a second system, measure operator time saved, and decide whether to expand or stop.

What this workbook is for

Use it as a build companion, not a passive ebook. Each section gives you a decision, a prompt, a concrete artifact to create, and a check that keeps the agent honest. Work through it with one real store question and keep the resulting repo under version control.

Your next useful agent is probably smaller than you think

Pick one recurring decision where inputs are known, the cost of delay is visible, and a human can review the recommendation. Load the skill that matches the platform. Prove the read-only path. Then earn the right to automate the action.

DOWNLOAD THE WORKBOOK

Keep this blueprint beside your repo.

getmisha.co/agent-launch

Need the system built with your team?

See Misha AI in action.

getmisha.co/demo

Sources and further reading

Claude Code: code.claude.com/docs/en/overview · Skills: code.claude.com/docs/en/skills · MCP: code.claude.com/docs/en/mcp · Ecommerce skills: github.com/finsilabs/awesome-ecommerce-skills · Current Claude pricing: claude.com/pricing

Prepared by Misha AI. Check current Claude pricing, documentation, and platform API terms before production use.