Build a self-improving GTM system.

A guide to building a GTM system for Claude Code, Codex, and other agents.

`context + workflows + guardrails + learning`

Why read this deck?

This deck explains how to build a GTM system that agents can use to run real sales and marketing work.

01

Understand the system

- how to give agents the customer and market context they need
- how to turn repeated GTM work into workflows
- feed reviewed learning back into the next run

02

Build your own

- create the directory structure
- write the first context pages
- create your first workflows

What is a GTM system?

A GTM system lets Claude Code and Codex orchestrate sales and marketing work.

1
context

What the system knows about your customers, market, offer, proof, and voice.

2
workflows

The repeatable GTM jobs the system can run, such as researching accounts, building campaigns, or reporting performance.

3
guardrails

Rules that stop the system from sending, spending, publishing, or changing records without approval.

4
learning

What gets saved after each run so the next run starts from better information.

Problem 01: Most teams lose what they learn.

Customer calls, ad playbooks, campaign metrics, and landing page tests create useful context. This is gold dust... but often stays scattered and unused.

customer calls

quotes, pains, objections

ad playbooks

angles, spend, winners

campaign metrics

conversion, CAC, signal quality

positioning notes

ICP, proof, message changes

landing page tests

claims, layouts, objections

problem

The next task starts with someone piecing the story back together: finding notes, remembering decisions, and working out which evidence still matters.

Solution 01: Build the context layer first.

Before touching any tools or automation, the system needs a clear picture of who the customer is, what they care about, what has worked, and how the company should speak.

collect

calls, CRM, reports, docs

distil

ICP, personas, proof, voice

link

claims back to evidence

read

agents start from context

update

reviewed learning returns

context layer

The context layer is the base of the system. You will create a ‘GTM wiki’: short, readable pages with clear names and links back to the evidence.

Problem 02: GTM work is manual.

Most GTM jobs are manual, time consuming (and painful)... leaving little room for creative thinking.

ICP analysis

same question, new spreadsheet

account enrichment

creating infinite CSVs

campaign planning

long briefs written by hand

outbound

same segment logic, new thread

reporting

digging through campaign managers

problem

Agents help most when the same task happens often enough to become a reusable workflow.

Solution 02: Package tasks into workflows.

A workflow should show how the work starts, what instructions it follows, what context it reads, which tools it can use, and where the output is saved.

command
human entry point

skill
saved procedure

context
important information

tools
which apis or services

output
reviewable files

review
approve and learn

workflow rule

A skill is the instruction file for a workflow. It tells the agent what to read, what to do, which tools it can call, and where to save the result.

Problem 03: Agents need boundaries.

GTM agents can touch budgets, outbound messages, CRM records, public pages, and customer-facing claims. Approval needs to be built into the system from the start.

ad spend

budgets, bids, launches

CRM writes

stages, owners, notes

outbound sends

emails, LinkedIn, sequences

public claims

ads, pages, social posts

problem

When approval is unclear, people either block the agent or let it take risks they cannot audit.

Solution 03: Put guardrails around live work.

Guardrails make the safe path explicit. The agent can read and draft freely, then ask before anything touches spend, customers, or production systems.

read freely

metrics
transcripts
files

draft safely

ads
emails
campaign specs

prepare carefully

CRM updates
campaign changes

approve explicitly

send
spend
publish

approval rule

Draft first. Review before live. Any change to spend, customers, CRM, or public pages needs a clear human approval step.

Problem 04: Agents stop working…

A useful agent can work well for one use case, then struggle when the ICP, channel, offer, tool, or success metric changes.

new use case

the old workflow misses steps

new evidence

the context stays stale

new channel rule

the guardrail is incomplete

new metric

success is measured differently

new positioning

old language keeps returning

problem

When learning only lives in chat history, the next run starts from old assumptions.

Solution 04: Create a learning loop.

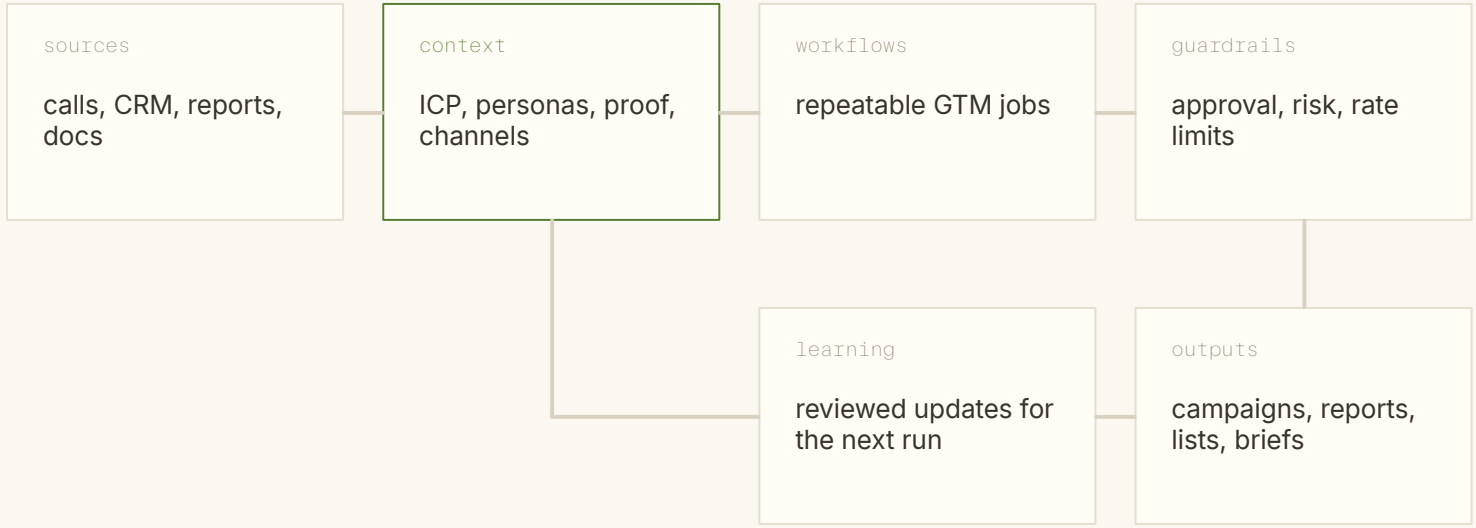
Create a ‘recursive loop’ where results and feedback auto-updates the context, workflow, guardrails, or checks. Each finished run can improve the next one.

learning rule

Self-improving means the system has a place to save what it learned.

The whole system is one learning loop.

Evidence becomes context. Workflows use that context. Guardrails control live actions. Reviewed outputs create learning.



the spine

This loop is the spine of the deck. Every file and workflow should make the next pass easier.

part 1

Map the system architecture.

Turn the model into a folder of plain files: sources, context, workflows, tools, outputs, checks, guardrails, and learning.

part 1 architecture

The system has six core parts.

A folder can hold the system when each part has a clear job.

sources

raw calls, CRM exports, reports, docs

context

current ICP, personas, voice, proof, channels

skills

repeatable workflows agents can run

tools

APIs, scripts, services, and local adapters

outputs

the work produced by each run

evals

checks for quality, facts, voice, and risk

Part 01: sources.

Sources are the raw material: calls, CRM exports, campaign reports, docs, research notes, and anything the system should be able to check.

sources	context	skills	tools	outputs	evals
raw evidence	wiki pages	saved workflows	safe access	saved work	quality checks

source folder

```
sources/
  calls/
  crm-exports/
  campaign-reports/
  support-threads/
  research-notes/
```

use sources to

- Check quotes, claims, and numbers.
- See where a belief came from.
- Rebuild context when the market changes.
- Keep useful evidence close to the work.

Part 02: context.

Context is what agents read before they act. It works best as short, named pages with clear links back to evidence.

context pages

```
context/icp.md
context/personas.md
context/positioning.md
context/proof.md
context/channel-linkedin.md
runner-instructions.md  -> read order
```

context should contain

- The current ICP and account logic.
- Personas, pains, objections, and proof.
- Voice, positioning, and channel rules.
- Enough source links to check important claims.

Part 03: skills.

A skill is a saved set of instructions for a repeated job. It says when to use it, which context to read, which tools are allowed, and what output to save.

sources	context	skills	tools	outputs	evals
raw evidence	wiki pages	saved workflows	safe access	saved work	quality checks

skill shape

```
persona-extraction/SKILL.md

read: transcripts, ICP, voice
do: pull quotes, extract jobs, write persona cards
tools: transcript search, filesystem
save: context/personas.md after review
```

use a skill when

- The same task repeats.
- The output should follow the same standard.
- Agents need clear boundaries before using tools.

Part 04: tools.

Tools are the connectors agents use to work with services and files: scripts, API wrappers, CLI commands, MCP servers, browser actions, or local adapters.

sources	context	skills	tools	outputs	evals
raw evidence	wiki pages	saved workflows	safe access	saved work	quality checks

example tools

```
hubspot.readAccounts
clay.enrichCompanies
apollo.findContacts
linkedin.readBenchmarks
filesystem.writeFile
```

approval rule

- Read-only tools can run freely.
- Drafting tools can write local files.
- Publishing, spend, and CRM writes need approval.

Part 05: outputs.

Outputs are the files created by each run. They make the work reviewable, reusable, and easier to compare with the next version.

sources	context	skills	tools	outputs	evals
raw evidence	wiki pages	saved workflows	safe access	saved work	quality checks

output folder

```
outputs/
  campaigns/
  reports/
  personas/
  account-notes/
  decisions/
```

good outputs include

- The finished asset or recommendation.
- The assumptions and source evidence.
- Open questions and approval notes.
- A short learning update when something changes.

Part 06: evals.

Evals are repeatable checks for quality, facts, voice, risk, and performance. They help decide whether the result should change context.

sources	context	skills	tools	outputs	evals
raw evidence	wiki pages	saved workflows	safe access	saved work	quality checks

eval files

```
evals/claims.md
evals/voice.md
evals/channel-rules.md
evals/risk.md
evals/performance.md
```

evals help you

- Catch weak claims before they ship.
- Keep tone and proof standards consistent.
- Review risky actions before they affect customers.
- Decide what should change before the next run.

Each workflow names what it needs.

Each workflow should name what to read, which tools it can use, what output to save, and which checks to run.

01 command	Names the job and the output required.
02 skill	Defines the workflow, read order, tool permissions, and output shape.
03 context	Loads the ICP, persona, voice, proof, and channel files for this job.
04 sources	Checks raw evidence before making claims or changing beliefs.
05 evals	Runs the quality, voice, fact, and risk checks before review.

where to write it

The read order should be visible in the file your agent reads first, such as CLAUDE.md, AGENTS.md, or runner instructions.

part 2

Build the context layer.

Turn customer evidence into the context pages agents read before doing GTM work.

part 2 context

Build 01: identify your ICP.

Pull revenue, margin, retention, expansion, and deal data before choosing the ICP the system should focus on.

value

revenue + retention

enrich

accounts + signals

language

calls + threads

angles

voice + proof

inputs

```
bookings.csv
deals.csv
revenue-by-account.csv
retention.csv
expansion.csv
margin.csv
```

output

- Rank the best customers by value.
- Cluster them by role, industry, size, and use case.
- Choose a primary and secondary ICP.
- Save the decision in context/icp.md.

Build 02: enrich top accounts.

Customer data shows which customers create the most value. Enrichment explains what those accounts have in common and which signals predict fit.

value	enrich	language	angles
revenue + retention	accounts + signals	calls + threads	voice + proof

tools and fields

```
Clay -> company size, industry, tech
Apollo -> roles and contacts
LinkedIn -> buyer profiles
Firecrawl -> website signals
AI research -> triggers and context
```

output

- A richer account profile.
- Signals that predict fit.
- Sharper personas and objections.
- A better target account list.

Build 03: pull language from transcripts.

Sales calls, support threads, interviews, and lost-deal notes show the jobs, anxieties, objections, and exact words customers already use.

value

revenue + retention

enrich

accounts + signals

language

calls + threads

angles

voice + proof

inputs

sales transcripts
support threads
customer interviews
lost-deal notes
call summaries

extract

- Jobs to Be Done and Four Forces patterns.
- Exact quotes customers use.
- Objections, anxieties, and pull factors.
- Disqualifying jobs and poor-fit signals.

Build 04: turn this information into ‘personas’.

The voice and messaging pages should turn customer evidence into reusable decisions: voice traits, proof standards, and angle ideas for ads, emails, and landing pages.

value

revenue + retention

enrich

accounts + signals

language

calls + threads

angles

voice + proof

angle matrix

```
persona | pain | proof | next angle

CFO | margin risk | saved hours | audit trail
Ops | manual handoff | faster flow | workflow
```

save into context

- context/voice.md
- context/proof.md
- context/personas.md
- context/angles.md

part 3

Build the workflows.

Turn the context layer into repeatable GTM flows.

part 3 workflows

Start with the GTM stack.

A workflow is one repeatable job inside the wider GTM system. The same architecture can support research, acquisition, conversion, measurement, and orchestration.

job		what it does	example tools	system output
01	**Research**	find and understand accounts	Clay, Apollo, Fireflies, web research	ICP, account notes, signals
02	**Acquire**	reach people and create demand	Meta, LinkedIn, outbound, partners	campaigns, lists, replies
03	**Convert**	turn interest into pipeline and revenue	HubSpot, CRM, calendar, call notes	pipeline, revenue, attribution
04	**Measure**	read performance and decide what changed	HubSpot, GA4, ad platforms, reports	scorecards, learnings, evals
05	**Orchestrate**	coordinate humans, agents, and tools	agent runners, approvals, logs	runs, approvals, learnings

Run path 01: command.

The command is how a person starts the run. It names the GTM job and gives the agent a predictable route through the system.

command	skill	context	tool calls	output	review
/new-linkedin campaign	linkedin-campaign SKILL.md	ICP, personas, channel rules	LinkedIn, HubSpot, Clay, Apollo	campaign pack + learning file	approve spend and publish

command

/new-linkedin-campaign

Run this when you want the system to draft a LinkedIn campaign from the current ICP, account list, positioning, proof, and channel rules.

what happens next

- The runner loads the LinkedIn campaign skill.
- The skill tells the agent which context to read.
- Tools enrich accounts and check channel details.
- The result is saved as a campaign pack for review.

Run path 02: skill.

The skill is the instruction file for the workflow. It keeps the campaign task consistent: what to read, what to do, which tools are allowed, and when to ask.

command	skill	context	tool calls	output	review
/new-linkedin campaign	linkedin-campaign SKILL.md	ICP, personas, channel rules	LinkedIn, HubSpot, Clay, Apollo	campaign pack + learning file	approve spend and publish

skill file

linkedin-campaign/SKILL.md

```
read: ICP, personas, target accounts, LinkedIn rules
do: choose segment, draft angle, write campaign pack
tools: HubSpot, Clay, Apollo, LinkedIn, filesystem
approval: draft freely; ask before spend or publish
```

why it matters

- The workflow lives in a file the team can reuse.
- The same campaign job can run again next week.
- Live changes stay behind explicit approval.

Run path 03: context.

The agent should know the customer, the segment, the proof, and the channel before it calls tools or drafts copy.

command	skill	context	tool calls	output	review
/new-linkedin campaign	linkedin-campaign SKILL.md	ICP, personas, channel rules	LinkedIn, HubSpot, Clay, Apollo	campaign pack + learning file	approve spend and publish

context read order

```
context/icp.md
context/personas.md
context/positioning.md
context/proof.md
context/channel-linkedin.md
data/target-accounts.csv
```

what this gives the agent

- Who the campaign is for.
- Which pains, objections, and proof matter.
- What a good LinkedIn campaign should do and avoid.
- Which accounts or segments are worth targeting first.

Run path 04: tool calls.

Tools are the connectors agents use to work with services and files: scripts, API wrappers, CLI commands, MCP servers, browser actions, or local adapters.

command	skill	context	tool calls	output	review
/new-linkedin campaign	linkedin-campaign SKILL.md	ICP, personas, channel rules	LinkedIn, HubSpot, Clay, Apollo	campaign pack + learning file	approve spend and publish

tool calls

```
hubspot.readAccounts
clay.enrichCompanies
apollo.findContacts
linkedin.readBenchmarks
linkedin.createDraftCampaign  # after approval
filesystem.writeCampaignPack
```

approval rule

- Read-only calls can run freely.
- Drafts can be written locally.
- Spend, publishing, and CRM writes need approval.

Run path 05: output and review.

The workflow ends with files the team can inspect: the campaign spec, the audience, the creative brief, the approval questions, and the learning update.

command	skill	context	tool calls	output	review
/new-linkedin campaign	linkedin-campaign SKILL.md	ICP, personas, channel rules	LinkedIn, HubSpot, Clay, Apollo	campaign pack + learning file	approve spend and publish

saved output

```
outputs/campaigns/linkedin/.../spec.md
audience.csv
creative-brief.md
approval-checklist.md
learning/channel-linkedin.md
```

review

- Check the audience, claims, proof, offer, and spend.
- Approve publishing only when the risks are clear.
- Save useful learning after review so the next run starts from better context.

part 4

Guard live actions.

Put boundaries around the parts of the system that can spend money, send messages, publish campaigns, or change customer records.

part 4 guardrails

Guardrail 01: approval rules.

GTM work touches live budgets, CRM records, outbound messages, and brand reputation. The approval rules should be clear from the start.

read freely

metrics
transcripts
files

draft safely

ads
emails
campaign specs

prepare carefully

CRM updates
campaign changes

approve explicitly

send
spend
publish

principle

Draft first. Review before live. Any change to spend, customers, CRM, or public pages needs a clear human approval step.

Guardrail 02: label what each tool can change.

Tools should say what kind of action they take. Reading a report is different from changing CRM data, sending outbound, or publishing ads.

read	draft	prepare	write	launch	audit
no approval	local files	staged API	approval	explicit	logs

tool changes

```
file_write
crm_write
email_send
external_post
ad_publish
llm_spend
git_push
```

guardrail works when

- Each skill says what it can read, draft, or change.
- Risky workflows do not run without a review step.
- Live changes route through approval.
- The person reviewing the run can see what changed and why.

Guardrail 03: respect rate limits.

External APIs have quotas and platform rules. The system should throttle requests, retry cleanly, stop on rate-limit errors such as 429s, and write logs to understand what happened.

read	draft	prepare	write	launch	audit
no approval	local files	staged API	approval	explicit	logs

rate-limit plan

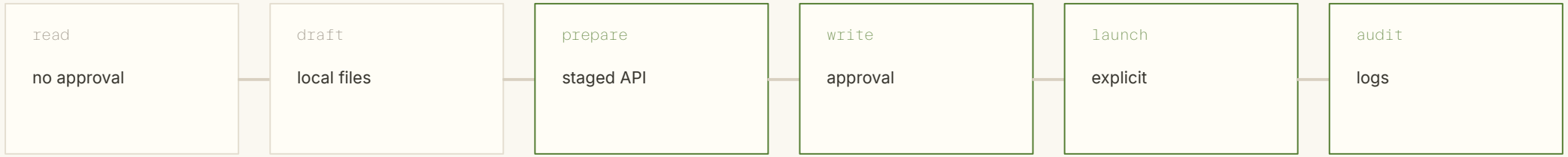
```
per app
per account
per user or member
retry-after handling
backoff and batching
request IDs in logs
```

guardrail works when

- Read each platform's current API limits.
- Batch and cache where the API supports it.
- Stop and ask if a live channel starts failing.
- Log 429s, request IDs, and retry decisions.

Guardrail 04: stage writes before launch.

Paid channels should create drafts or paused campaigns first. Turning something live should be a separate approved step, with IDs and request logs saved after the run.



staged launch

```
.growthOS/approvals.jsonl
status: PAUSED or DRAFT
activation: separate approval
outputs/launches/.../id-map.md
logs/adapter-events.jsonl
```

guardrail works when

- Approval payload is visible before execution.
- Create calls cannot start spend by accident.
- The ID map is saved for audit and rollback.
- Sandbox tests prove the live path is gated.

part 5

Make the system learn.

Turn reviewed outputs into better context, examples, rules, and evals.

part 5 learning

Reviewed outputs improve the next run.

A run produces output. A human reviews what changed. The approved learning updates context, examples, rules, or evals.

what improves

Small, sourced changes make the system better without making it hard to audit.

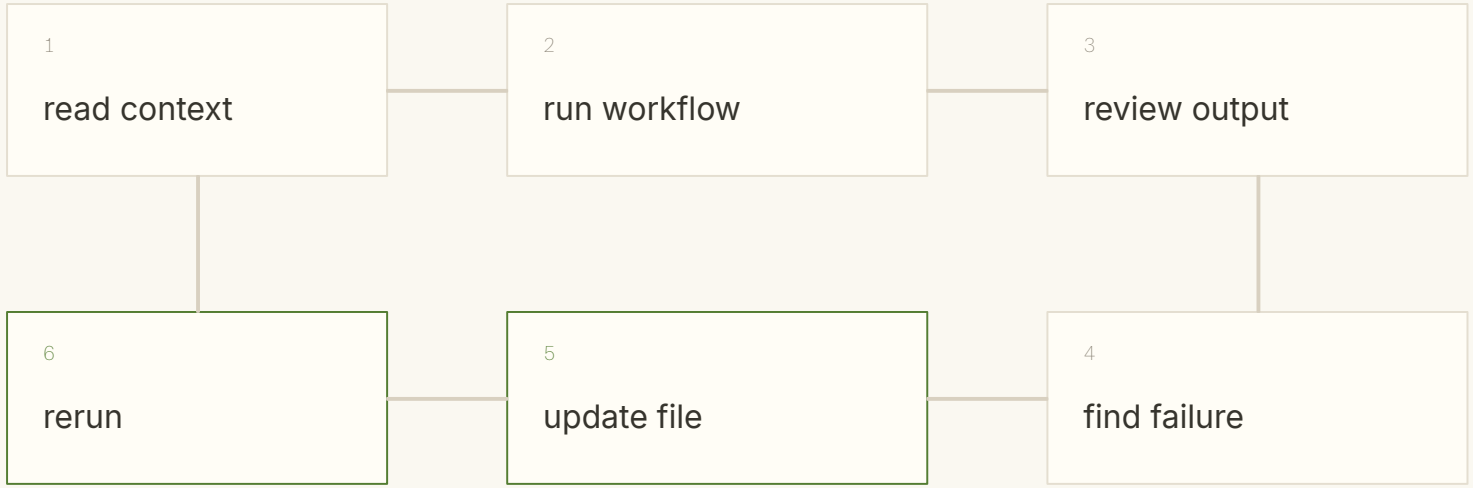
The learning lives in files.

Each useful run should leave a small, reviewable update. The update should say what changed, why it changed, and which evidence supports it.

`outputs/<run>/report.md`	what happened in the run
`outputs/<run>/learning.md`	what the system should remember
`context/<page>.md`	approved updates to ICP, proof, voice, or channel logic
`evals/<check>.md`	new checks for quality, facts, risk, or performance
`skills/<workflow>/SKILL.md`	workflow changes only when the process itself improves

Use the loop to improve workflows.

Use the Karpathy wiki pattern to keep context readable: update the page that would have changed the result, then rerun the workflow.



keep it small

Change one thing at a time: context, skill instructions, tool behavior, or eval criteria.

Checks decide what should change.

Checks help decide whether the result was good, whether the context was wrong, and what should change before the next run.

output	review	eval	learning	context	skill
campaign pack	human notes	quality checks	approved update	better next run	only if needed

eval questions

```
Is the claim supported?
Does this match our voice?
Did the workflow use the right context?
Was the result useful?
What should change before the next run?
```

good learning updates

- Name the source.
- Explain the reason.
- Update the smallest file possible.
- Leave the next run easier to audit.

part 6

Getting started.

Build the first useful version in order. Each step points to the prompt pack and ends with a review gate before you move on.

getting started

Start 01: create the folder.

The folder is the system. Start by making the core parts visible, named, and easy for Claude Code, Codex, or another agent to read.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

link: prompt-pack.md#01-create-folder

creates:
runner-instructions.md
sources/ context/
skills/ tools/
outputs/ evals/

move on when

- The six core folders exist.
- The instruction file names what to read first.
- The agent can explain what each folder is for.
- Open questions are written down.

Start 02: build the context layer.

Use existing evidence before asking new questions: customers, transcripts, CRM exports, support threads, campaign reports, and website copy.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

link: prompt-pack.md#02-build-context

writes:
context/index.md
context/icp.md
context/personas.md
context/proof.md
context/voice.md
context/channel-notes.md

move on when

- The best customer patterns are clear.
- Personas use customer language.
- Important claims point back to evidence.
- Context is short enough to read first.

Start 03: add one workflow.

Pick one repeated GTM job and make it reliable before connecting more channels. The first workflow should read context, produce output, and ask before risky actions.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

link: prompt-pack.md#03-add-workflow

writes:
skills/<workflow>/SKILL.md
outputs/<workflow>/
evals/<workflow>.md

move on when

- The skill defines read order and output shape.
- The workflow runs from sample input.
- The output is saved in the expected place.
- The review checklist catches weak work.

Start 04: connect one safe tool.

Begin with read-only access. Let the agent inspect accounts, transcripts, reports, or campaign data before you allow it to write to live systems.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

```
link: prompt-pack.md#04-connect-tool

writes:
tools/<tool>.md
tools/<tool>.mjs
outputs/tool-test.md

read first; write later
```

move on when

- The tool has clear inputs and outputs.
- A read-only test run succeeds.
- Errors and rate limits are logged.
- Write actions are blocked until approved.

Start 05: add measurement and guardrails.

The first version needs just enough structure to know whether the workflow helped and whether any risky action is still behind approval.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

```
link: prompt-pack.md#05-measure-and-guard

writes:
evals/performance.md
evals/voice.md
evals/risk.md
.growthOS/approvals.jsonl
outputs/reports/
```

move on when

- The scorecard says what good means.
- Approval rules cover spend, sends, and CRM writes.
- The report format can be repeated.
- The system knows what requires human review.

Start 06: run one loop and audit it.

The first useful version passes when one workflow runs end to end, saves a reviewable output, and leaves the next run with better context.

folder	context	workflow	tool	measure	run
file system	wiki pages	one skill	safe access	scorecard	audit loop

prompt and output

```
link: prompt-pack.md#06-run-and-audit

runs:
/new-linkedin-campaign
-> output saved
-> report written
-> learning reviewed
-> context updated
```

move on when

- A user can inspect one real output.
- The report explains what happened.
- The learning note has a source and a reason.
- The audit names the smallest next tweak.

A good system makes the next run easier.

Each run leaves behind the files the next run needs: context, workflows, tools, approvals, outputs, checks, and learning.

- Humans can read it and understand the strategy.
- Agents can read it and do the next useful task.
- The build can be reviewed step by step before it expands.
- Every run leaves the system with a better starting point.

Build your own.

Use the deck to understand the model, the prompt pack to create the first version, and the folder to keep improving it.

omarismail.com/projects/growth-os