UPDATED EDITION — FOUNDER'S PLAYBOOK

TM

THE ZERO-API STACK

Build A Full AI Team. Run It For Almost Nothing.

A founder's playbook from someone who actually built it.

$1,847 → $68/mo
Real monthly cost

70%
Runs locally for free

4 AI roles
One complete team

Tavon Maven · @tavon.maven · tavonmaven.com · Verified Creator

BEFORE YOU DIVE IN

2%

of people building with AI actually know this.

Most are paying for compute they do not need,
running on infrastructure they do not control,
and wondering why their AI costs keep climbing.

TM

About the author

I am Tavon Maven. Building with AI since early 2024 --
before most founders knew what a token was.

I run Platos, an AI agency. My stack: Claude, Cowork,
Gemini, Codex, and Ollama -- each doing what it does best.

Real clients. Real money. $1,847/mo API bill cut to $68/mo.
This playbook is the exact system I run in production today.

Since 2024	Platos Agency	Real production
Building with AI	Founded	Real clients

What follows is not theory. It is the exact system I run in production today.

Let’s get into it.

00 / CONTENTS

What’s Inside

UPDATED EDITION: What changed — Removed n8n. Added Cowork, Tasks, Claude Code, MCP.
New framework: 4 AI team roles. All agent templates rebuilt for the 2025 Claude stack.

1 **Why I Stopped Paying API Bills** p.4
The $1,800 wake-up call that changed everything.

NEW **Era 1 vs Era 2** p.5
Why n8n is dead and prompting won. Steal this framing.

NEW **The AI Team Framework** p.6
4 roles. One stack. Runs for almost nothing.

2 **The Multi-Model Orchestra** p.7
Which tool for which job after 18 months of production.

3 **Your Machine as the Server** p.8
Your MacBook is already a GPU server. Here is the proof.

4 **The 15-Minute Setup** p.9
Ollama + Claude Code running in one terminal session.

5 **Running in Production** p.10
Beyond your laptop: VPS, Groq, Together.ai.

6 **Model Selection Guide** p.11
Which model for which job after testing 12 of them.

7 **Agent Template 1: Content Repurposer** p.12
1 piece of content into 12 assets for $0.

8 **Agent Template 2: Lead Qualifier** p.13
Score every lead before it hits your CRM.

9 **Agent Template 3: Internal Ops Bot** p.14
200 internal questions answered daily for $0.

10 **Cowork + Ollama: My Silent COO** p.15
Scheduled orchestration with free local inference.

NEW **Claude Code: Your Always-On Engineer** p.16
Reads your whole codebase. Writes, runs, deploys.

NEW **MCP Servers: No More Glue Code** p.17
Native tool connections without writing integrations.

11 **Workflow Template: End-to-End Example** p.18
A real workflow I run every Friday for a client.

12 **Hybrid Architecture: When to Use APIs** p.19
The honest 30% that still needs frontier models.

13 **The Tool-Switching Decision Tree** p.20
8 heuristics from 18 months of production builds.

14 **Mistakes I Made** p.21
Five failures so you go straight to what works.

15 **The Bridge** p.22
If you would rather have this built for you.

16 **BONUS: The Zero-API Prompt Library** p.23
12 prompts tagged with which tool runs them best.

The Month My API Bill Hit $1,800

I was running three clients on Claude and GPT-4. Summarization pipelines, document processors, a chatbot that answered the same 40 questions 10,000 times a day.

Then I got the invoice.

$1,847 in one month. For a side project. That was the wake-up call.

Here is what I learned after 3 months of switching everything I could to local models:

- → 70% of what I was using GPT-4 for could be done by a 7B model running on my laptop
- → The remaining 30% genuinely needed a frontier model -- complex reasoning, nuanced writing, agents
- → My real cost dropped to ~$68/month running Claude, Cowork, Gemini, Codex, and Ollama as a full stack

> *The 70/30 Rule: 70% of AI tasks can run locally for $0. Save the frontier model budget for the 30% where it genuinely matters.*

The goal of this playbook is not to go fully off-grid from OpenAI and Anthropic. It is to be intentional about which tasks actually need the big models, and run everything else locally for free.

$1,847/mo	$1,779/mo	$68/mo
API Bill (Before)	Monthly Savings	Current Cost

Two Eras of AI Automation

Era 1 (2019-2023)

Zapier · Make · n8n

Assumption:
Non-technical people can't code

Friction:
Learning the tool itself takes weeks

Ceiling:
Still hits a wall at complex logic

Era 2 (2024+)

Cowork · Tasks · Claude Code

Assumption:
People CAN describe what they want

Friction:
The interface IS natural language

Ceiling:
Bounded by your operational knowledge

The key insight:
n8n abstracted code into drag-and-drop. Prompting abstracted drag-and-drop into language. It is not a better n8n -- it is the end of the category.

Non-technical people could not code. They CAN describe what they want. That is all prompting is -- precise description. The skill gap did not disappear. It moved.

Era 1 required learning a tool. Era 2 requires knowing your business well enough to describe it. That is why your workflows are better than someone who just learned to prompt last week. Experience is the new technical skill.

The unlock: you do not need to learn how to code, drag nodes, or write YAML. You need to know your operation well enough to describe it precisely. That is the only skill that compounds.

Questions about this? → @tavon.maven on Instagram

Your AI Team:
4 Roles. One Stack.

Every task in your business maps to one of four AI roles. Match the task to the role. Run the role on the cheapest tool that delivers acceptable quality.

Ollama · Llama 3.1 8B · Mistral · Phi-3 Mini

Does the heavy lifting. Summarization, classification, RAG, routing, drafts. Runs 24/7 on your machine. Never sends data anywhere.

ROLE 1
THE COMPUTE LAYER
$0/month
LOCAL

Claude Opus · GPT-4o · Gemini Ultra

The senior advisor. Complex reasoning, nuanced writing, multi-agent chains, anything client-facing that must be perfect. Use sparingly and deliberately.

ROLE 2
THE INTELLIGENCE LAYER
< $50/month
FRONTIER

Claude Cowork · Tasks · Claude Code

Your COO. Schedules work, delegates to the right tool, manages files, reports back. Replaces n8n, Zapier, and half your manual workflows.

ROLE 3
THE OPERATIONS LAYER
~$20/month
ORCHESTRATION

Claude Code · Codex · Cursor

Your engineer. Reads your entire codebase, writes integrations, debugs, deploys. Connects to MCP servers for native tool access without glue code.

ROLE 4
THE DEVELOPMENT LAYER
Subscription
DEVELOPMENT

Push every task to the cheapest role that delivers acceptable quality. Only escalate when quality actually requires it. This single discipline is what dropped my bill from $1,847 to $68.

The Multi-Model Orchestra

Stop trying to make one model do everything. After 18 months in production, here is exactly which tool I reach for and why:

Job	Tool	Why
Frontend code (React, Tailwind, UI)	**Claude**	Cleanest JSX. Best taste. Reads design intent.
Backend, infra, edge cases	**Codex**	Faster on boilerplate. Better with weird stack traces.
Long-context research, doc dumps	**Gemini**	1M+ context. Cheap. Handles 200-page PDFs.
Task delegation, scheduling, files	**Claude Cowork**	Runs tasks, sends messages, manages files. My COO.
High-volume internal inference	**Ollama (Llama 3.1 8B)**	Free. Private. Fast enough. Lives on your Mac.
Routing, intent, sub-second calls	**Phi-3 Mini (local)**	2 GB model. Answers in <500ms.
Final client-facing copy	**Claude (frontier)**	The polish layer. Worth every token.

My honest monthly split:

~60% of inference	Ollama (local, free)	**$0**
~15%	Claude Cowork	**$20/mo**
~10%	Gemini 1.5 Flash (long-context)	**~$8/mo**
~10%	Claude / Codex (frontier)	**~$40/mo**
~5%	Phi-3 routing (local)	**$0**

Total: ~$68/month running an agency-sized AI operation.

The rule: stop asking which AI is best. Ask which is the cheapest tool that gets this job to acceptable quality.

Questions about this? → @tavon.maven on Instagram

Your Laptop Is Already A GPU Server

A modern MacBook with Apple Silicon (M1/M2/M3) runs a 7B parameter model in real time. Fast enough for production. Free forever.

On Windows with an NVIDIA GPU (even a 3060), you are running 13B models at respectable speeds.

→ Ollama -- the local model runner
→ A model -- start with Llama 3.1 8B or Mistral 7B
→ Claude Code -- your agentic engineer (covered on p.16)

Hardware	RAM	Best Model	Speed
MacBook M1/M2/M3	16 GB	7B - 13B	Fast Fast
Windows + NVIDIA 3060/3070	12 GB VRAM	7B - 13B	Fast Fast
Linux VPS (CPU only)	32 GB RAM	7B	Slow (works) Slow (works)

I run everything on an M2 MacBook Pro with 16 GB RAM. Go-to model for 90% of tasks: Llama 3.1 8B. Responds in ~4 seconds. Clients cannot tell the difference.

The 15-Minute Setup

1 Install Ollama

```
brew install ollama
# or download from ollama.ai
```

Available on Mac, Windows, Linux.

2 Pull Your First Model

```
ollama pull llama3.1
```

~4.7 GB. Runs on any machine with 8 GB+ RAM.

3 Test It In The Terminal

```
ollama run llama3.1
# Type: Summarize this in 3 bullet points: [text]
```

If you get a response, you are live.

4 Access Via API (OpenAI-compatible)

```
curl http://localhost:11434/api/generate \
  -d '{"model":"llama3.1","prompt":"Hello"}'
```

Port 11434. Drop-in replacement for the OpenAI SDK.

5 Swap Into Existing Code -- One Line

```
# Before (OpenAI -- paid)
client = OpenAI(api_key='sk-...')

# After (Ollama -- free)
client = OpenAI(base_url='http://localhost:11434/v1',
                api_key='ollama')
```

One line change. That is it.

6 Install Claude Code

```
npm install -g @anthropic-ai/claude-code
# Then in any project folder:
claude
```

Questions about this? → @tavon.maven on Instagram

07 / PRODUCTION

Taking It Beyond Your Laptop

Running locally is great for prototyping. For production apps that need to run 24/7, you have two options:

Option 1

Self-Host on a VPS

A Hetzner CPX31 (4 cores, 8 GB RAM) costs $8.50/month. Run Ollama on it. API calls go to your own server. I have had clients running on this for 6 months straight.

$8.50 - $20/month

Option 2

Groq or Together.ai

Both offer free API tiers for Llama and Mistral. Groq is insanely fast -- 500 tokens/second. Use for burst traffic or when you do not want to manage infrastructure.

$0 on free tier

For most small apps under 1,000 daily users: a $10/month Hetzner VPS running Ollama beats a $300/month OpenAI bill every time.

	Local / Ollama	OpenAI API
Cost	$0 - $20/mo	$150 - $1,800/mo
Data privacy	Fully local	Sent to OpenAI
Speed	2-6 s/response	0.5-3 s/response
Reliability	You own uptime	99.9% SLA

Which Model For Which Job

Not all local models are equal. Here is my honest take after testing 12 of them:

Model	Size	Best For	Avoid For	Rating
Llama 3.1 8B	4.7 GB	General tasks, summarization, Q&A	Complex code, nuanced writing	4 / 5
Llama 3.1 70B	40 GB	Everything frontier does -- free	Requires beefy hardware	5 / 5
Mistral 7B	4.1 GB	Fast responses, simple tasks	Long-context tasks	4 / 5
Qwen2.5 32B	20 GB	Code generation, reasoning	Light hardware	5 / 5
Phi-3 Mini	2.3 GB	Edge, ultra-fast routing	Anything complex	3 / 5
Gemma 2B	1.6 GB	Intent detection, classification	Open-ended generation	3 / 5

My current daily stack:
Llama 3.1 8B for client-facing summarization · Qwen2.5 32B for any coding task · Phi-3 Mini for webhook routing and intent detection

Template 1: The Content Repurposer

Turn 1 long-form piece into 12 assets. Zero API cost.

SYSTEM PROMPT TEMPLATE

```
You are a content repurposing assistant. Given a long-form piece, extract:
1. A 10-tweet thread (hook + 8 insights + CTA)
2. A LinkedIn post (200 words, 3 key takeaways)
3. An email newsletter summary (150 words, conversational)
4. 3 short-form video scripts (60 seconds each)
Maintain the author's voice. No fluff. Label each format clearly.
```

SAVINGS MATH

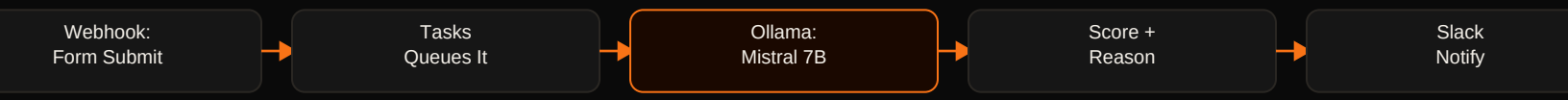
GPT-4o cost per run: ~$0.08
Ollama (Cowork-orchestrated) cost: $0.00
50 pieces/month: $4 vs $0
Annual saving: ~$48 (scales with volume)

Run with: Ollama -> Llama 3.1 8B ($0)

Questions about this? → @tavon.maven on Instagram

Template 2: The Lead Qualifier

Score every inbound lead before it hits your CRM.



SYSTEM PROMPT TEMPLATE

```
You are a lead qualification assistant for a B2B company.
Score the lead 1-10 based on:
  Budget signals (> $500/mo = +3)
  Timeline (immediate = +2, vague = 0)
  Company size (10-200 employees = +2)
  Problem clarity (specific pain = +2)
  Red flags (student/competitor/no budget = -3)
Output: Score, Tier (Hot/Warm/Cold), 2-sentence reason.
```

SAVINGS MATH

GPT-4o cost per lead: ~$0.04
Ollama + Tasks cost: $0.00
500 leads/month: $20/mo vs $0
Annual saving: $240

Run with: Ollama -> Mistral 7B ($0)

Questions about this? → @tavon.maven on Instagram

Template 3: The Internal Ops Bot

Answer 200 internal questions a day without burning API credits.

SYSTEM PROMPT TEMPLATE

```
You are an internal operations assistant. You have access to company
SOPs, FAQ docs, and the employee handbook. Answer accurately
and concisely. If the answer is not in the documents, say:
  'I do not have that -- please ask [manager name].'
Never guess. Never hallucinate policy.
```

RAG setup: chunk SOPs into 500-token segments, embed with nomic-embed-text (free via Ollama), store in ChromaDB (free, local), retrieve top-3 chunks per query. Setup time: ~2 hours.

SAVINGS MATH

GPT-4 Turbo for 200 queries/day: $60/month
Ollama + local RAG: $0/month
Annual saving: $720 -- data never leaves your machine

Run with: Ollama -> Llama 3.1 8B ($0)

Questions about this? → @tavon.maven on Instagram

Cowork + Ollama:
My Silent COO

Cowork schedules tasks, runs shell commands, manages files, and messages you when things finish. Ollama runs locally and serves an OpenAI-compatible API on port 11434. Connect them: scheduled orchestration plus free inference plus file system access in one loop. No API bill. No data leaving your machine.

CLAUDE COWORK (the operator)

- Schedules: every Mon 7am, run lead scoring
- Sends: Slack / iMessage / email
- Manages files: organizes outputs, versions docs
- Calls out via shell + API to Ollama

↓

OLLAMA (the compute) · $0/run

Llama 3.1 8B · Mistral 7B · Phi-3 Mini · Lives on your machine

One real workflow -- every morning at 7am:

1. Cowork scheduled task fires
2. Pulls overnight form submissions from synced CSV folder
3. Shells out to local Ollama: Score this lead 1-10, output JSON
4. Filters scores 7+ -- hot leads flagged
5. Drafts personalized first-touch reply in your voice
6. Drops replies in review-and-send folder + iMessages you a summary
7. Cold leads filed in Airtable for nurture

A VA doing this manually: ~$2,000/mo
Zapier + OpenAI alternative: ~$300/mo
Cowork + Ollama: $20/mo subscription + $0 inference = $20 total

Cowork is not a chatbot. It is a dispatcher. Ollama is not a toy. It is free compute. Together they are the cheapest ops team you will ever hire.

Claude Code:
Your Always-On Engineer

Claude Code is an agentic CLI tool that runs in your terminal. It reads your entire codebase, writes and edits files, runs shell commands, connects to external tools via MCP servers, and works autonomously on tasks you describe in plain English.

It is not a code autocomplete. It is an engineer that does not need you to supervise every keystroke.

INSTALL

```
npm install -g @anthropic-ai/claude-code
# Navigate to any project folder, then:
claude
```

What it does that nothing else does:

Reads your full codebase
Not just the file you open. It loads your entire repo as context. Finds the bug you described across 40 files.

Writes and runs code autonomously
You say: add authentication to this Express app. It writes the code, runs the tests, fixes the errors, tells you when it is done.

Connects to MCP servers
Native integration with Airtable, Notion, GitHub, Slack, and more. No glue code. No API wrappers. More on the next page.

Works with Ollama
Point Claude Code at your local Ollama instance for the heavy lifting. Claude handles reasoning. Ollama handles volume. $0 compute.

Full docs: claude.ai/code

Questions about this? → @tavon.maven on Instagram

MCP Servers:
No More Glue Code

MCP (Model Context Protocol) is Anthropic's open standard for connecting AI models directly to external tools and data sources. Instead of writing a custom API wrapper every time you want Claude to talk to Notion, GitHub, or Airtable -- you just add the MCP server and Claude connects natively.

This is what killed most of the n8n use case for technical workflows.

HOW IT WORKS

```
# Add to your Claude Code config (~/.claude.json):
{
  "mcpServers": {
    "filesystem": {"command": "npx", "args": ["@modelcontextprotocol/server-filesystem", "."]},
    "github": {"command": "npx", "args": ["@modelcontextprotocol/server-github"]},
    "airtable": {"command": "npx", "args": ["airtable-mcp-server"]}
  }
}
```

MCP servers worth installing today:

Server	Install	What It Replaces
Filesystem	`npx @modelcontextprotocol/server-filesystem`	Read/write any file on your machine. Claude organizes your outputs
GitHub	`npx @modelcontextprotocol/server-github`	Create PRs, read issues, push code. Your whole GitHub workflow by prompting.
Airtable	`npx airtable-mcp-server`	Read and write Airtable bases directly. Replaces the Airtable n8n node.
Notion	`npx @notionhq/notion-mcp-server`	Read/write Notion pages and databases. Replaces copy-pasting into Notion.
Slack	`npx @modelcontextprotocol/server-slack`	Send messages, read channels, post updates without a webhook workflow.
PostgreSQL	`npx @modelcontextprotocol/server-postgres`	Query your database in plain English. No SQL required.

Full MCP registry: modelcontextprotocol.io

MCP is what makes Claude Code feel like a teammate, not a tool. It reads your files, writes to your database, posts to your Slack, and pushes to your GitHub -- all from a single prompt.

Questions about this? → @tavon.maven on Instagram

Real Workflow: The Async Client Update Bot

Here is one I actually run for a client. Every Friday at 9am, fully automated:

1. Cowork scheduled task fires at 9am Friday
2. Cowork fetches last 7 days of project notes from Notion via MCP
3. Ollama (Llama 3.1 8B): Summarize these notes into a 200-word client update. Wins + blockers + next steps. Client-friendly tone.
4. Cowork formats output with client name and project header
5. Gmail MCP node sends the update to the client
6. Slack MCP notifies internal team that update was sent

"This runs every week. Never missed. Costs: $0 inference. Setup time: 3 hours. Client thinks I have a full operations team."

GPT-4o alternative: ~$0.03/run x 52 weeks = $1.56/year

Ollama cost: $0/year

Real value: Never forgetting to update a client again.

The Honest Truth:
Some Things Still Need Claude

I said 70/30 earlier. Here is what that actually looks like in a real production system:

Does output quality directly affect revenue or client perception?

YES → Frontier Model

Local Model ← NO

FRONTIER API (Paid)

- Final client-facing copy
- Complex multi-step reasoning
- Code review and refactoring
- Nuanced negotiation scripts
- One bad output = real consequences

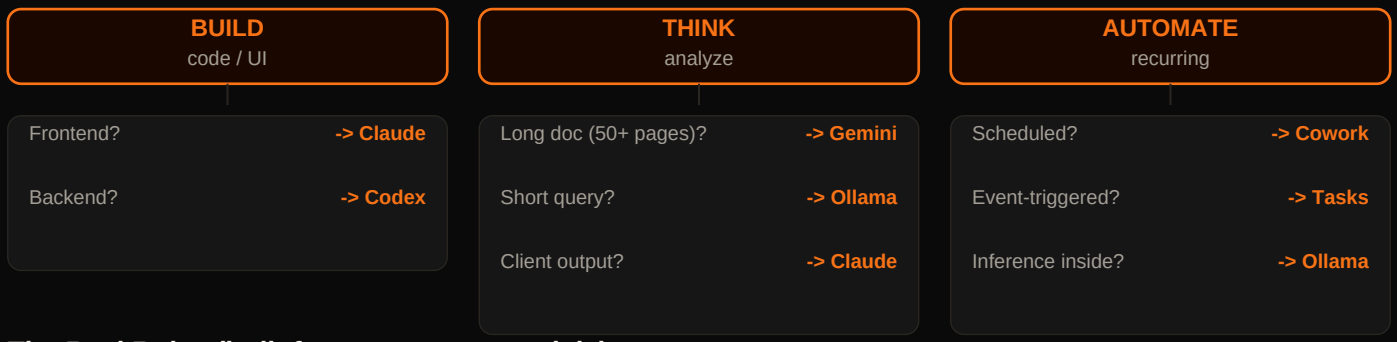
LOCAL MODEL (Free)

- Summarization
- Classification
- RAG retrieval
- First drafts
- Internal Q&A
- Lead scoring
- Routing / intent

The discipline is asking: does this task actually need the best model?
Usually the answer is no. And that one question saved me $1,779/month.

The Tool-Switching Decision Tree

What runs in my head in the moment. Steal it.



The Real Rules (built from reps, not tutorials):

→ Frontend = Claude. Codex writes working code but ugly components. Claude has taste.

→ Backend + infra = Codex. Handles cryptic errors and weird env issues.

→ >50 pages of docs = Gemini. Do not chunk when Gemini eats the whole thing.

→ <30 pages = skip RAG. Stuff it in context. RAG only pays off when docs will not fit.

→ Same job >3x per week = Cowork. If you do it a fourth time manually, you are losing money.

→ Volume >500/month = local model. Anything that hot belongs on Ollama.

→ Client sees the output = frontier model. $0.04 is cheap insurance.

→ Sub-second routing = Phi-3 Mini. 2 GB model, 500ms, perfect for intent detection.

Stop asking which AI is best. Ask: what is the cheapest tool that gets this to acceptable quality? Push every task down the stack until quality breaks.

5 Mistakes I Made
So You Skip Them

MISTAKE 1

Running 70B models on a 16 GB machine

LESSON

Start with 7B or 8B. They are 80% as capable for 20% of the hardware. I wasted 2 weeks chasing performance I did not need.

MISTAKE 2

Using local models for client output without a quality gate

LESSON

Always add a validation step. If score < 7/10, escalate to frontier automatically. Never ship raw local output to clients.

MISTAKE 3

Not versioning my prompts

LESSON

Treat prompts like code. Store them in git. Tag every working version. I have broken production workflows by improvising.

MISTAKE 4

Building integrations instead of using MCP

LESSON

I wrote custom API wrappers for Notion, GitHub, and Airtable before MCP existed. Now they are all one config file. Check the

MISTAKE 5

Not using Claude Code for onboarding new projects

LESSON

Claude Code reads an entire unfamiliar codebase in seconds. I wasted hours orienting myself manually. Now the first thing I do

If You Want This Built For You

This playbook gives you everything you need to build this yourself. If you have the time and enjoy the setup, go for it.

But if you are a founder and your time is better spent on product, sales, or customers -- I build this for people.

I take your current AI stack (or your Lovable/Bolt/Cursor app burning API credits) and rebuild the architecture: Ollama for local compute, Cowork and Tasks for orchestration, Claude Code for development, MCP for native integrations.

Most projects take 7-14 days. You end up with:

→ A local model stack running on your VPS or Mac
→ Cowork + Tasks workflows replacing your n8n and Zapier
→ Claude Code + MCP connecting natively to all your tools
→ A clear decision tree for what runs where and why

Here is how to reach me:

Bug Bounty
1-3 bugs fixed. 72-hour turnaround. Async.

DM me on Instagram →

★ MOST POPULAR
Production Pass
7-day full rebuild. Vibe-code to production.

Book a call →

AI Agent Upgrade
Custom agent + hybrid architecture. 10-14 days.

Book a call →

instagram.com/tavon.maven · tavonmaven.com

The Zero-API Prompt Library

12 prompts tagged with which tool runs them best. Screenshot this page.

The Summarizer

```
Summarize in exactly 5 bullets. Each sentence under 20 words. No intro or conclusion. Start immediately: [TEXT]
```

Run with: Ollama -> Llama 3.1 8B

The Classifier

```
Classify into ONE category from [CATEGORIES]. Output only the category name. No explanation. Text: [TEXT]
```

Run with: Ollama -> Mistral 7B

The Lead Scorer

```
Score 1-10. Output: Score, Tier (Hot/Warm/Cold), one-sentence reason. Nothing else. Inquiry: [TEXT]
```

Run with: Ollama -> Mistral 7B

The Tone Rewriter

```
Rewrite in [TONE] tone. Keep all facts. Same length. No opinions added. Original: [TEXT]
```

Run with: Ollama -> Llama 3.1 8B

The Extractor

```
Extract all [ENTITY TYPE] from this text. Output as JSON array. Nothing else. Text: [TEXT]
```

Run with: Ollama -> Qwen2.5

The FAQ Generator

```
Generate 5 FAQ Q&As from this product description. Each answer under 50 words. Format Q:/A: Product: [TEXT]
```

Run with: Ollama -> Llama 3.1 8B

The Email Subject

```
Write 5 subject lines. Each under 9 words. No emojis, no clickbait. Email: [TEXT]
```

Run with: Ollama -> Mistral 7B

The Intent Detector

```
Detect intent. Output ONE word: question/complaint/purchase-intent/churn-risk/support/spam. Message: [TEXT]
```

Run with: Ollama -> Phi-3 Mini

The First Draft

```
Write a [FORMAT] about [TOPIC] for [AUDIENCE]. Tone: direct, no fluff. Length: [WORD COUNT]. Start immediately.
```

Run with: Cowork + Llama 3.1 70B

The Feedback Analyzer

```
Analyze feedback. Output: Sentiment, Top 3 themes, Urgency (low/med/high). Under 100 words. Feedback: [TEXT]
```

Run with: Ollama -> Llama 3.1 8B

The Decision Framer

```
Decision: [DECISION]. List top 3 FOR and top 3 AGAINST. Be specific. No recommendations.
```

Run with: Cowork + Qwen2.5 32B

The Long Doc Summary

```
Summarize this document. Output: 5 key insights, 3 action items, 1 quote worth keeping. Doc: [TEXT]
```

Run with: Gemini (long-context)

Questions about this? → @tavon.maven on Instagram

TM

Tavon Maven

AI Builder · Production Engineer · Phoenix, AZ

instagram.com/tavon.maven · tavonmaven.com