The $0 DIY "Grok Bot" Blueprint

A step-by-step technical guide to building autonomous AI workflows.

SuperGrok Plus and similar premium tier agents charge upwards of $100/month for convenience. But underneath the slick UI, they are relying on basic API integrations, webhooks, and headless browsers. You can orchestrate the exact same workflows for $0 using self-hosted and free-tier tools.

The Core Stack: Make.com (Webhooks), n8n (Orchestration), Groq API (Inference), and Puppeteer (Browser execution).

Module 1: The Autonomous Inbox Manager

This module replaces premium "email assistant" bots. It listens for new emails, extracts the context, generates a highly contextual reply using open-source models, and saves it directly to your drafts folder.

Step 1: Set up the Make.com Trigger

Create a free Make.com account. Create a new scenario and add the **Gmail: Watch Emails** module. Set it to watch the *Primary* inbox. This acts as your event listener. The webhook will pull the sender's address, subject line, and plain-text body.

Step 2: Process via Groq API (Llama 3)

Add an HTTP module to your Make scenario to make a POST request to Groq. We use Groq because its LPU inference engine returns results in milliseconds for free.

URL: `https://api.groq.com/openai/v1/chat/completions`

```
{
  "model": "llama3-70b-8192",
  "messages": [
    {
      "role": "system",
      "content": "You are a highly efficient administrative assistant. Read the
provided email and draft a professional, concise reply."
    },
    {
      "role": "user",
      "content": "Subject: {{Subject}}
Body: {{TextContent}}"
    }
  ]
}
```

Step 3: Push to Drafts

Add a final **Gmail: Create a Draft** module. Map the output from the Groq HTTP module (specifically the `choices[0].message.content` node) into the body of the email draft, and map the sender's email to the "To" field.

Module 2: Headless Web Research (n8n + Puppeteer)

Premium bots brag about "living on their own computer." You achieve this by running a headless browser in a Dockerized n8n instance.

Step 1: Deploy n8n

If you have a local server or a cheap VPS, deploy n8n via Docker. It's completely free for self-hosting and doesn't limit your execution counts like cloud platforms.

> **Step 2: Configure the HTTP Request Node**
>
> Instead of standard API calls, route your target URLs through an open-source scraper like **Browserless** or set up a custom Puppeteer script in an execute node. This allows the workflow to bypass basic JS rendering barriers and pull raw HTML/Text.

> **Step 3: Schema Mapping and Export**
>
> Pass the scraped text to a local or free-tier LLM and force a JSON output schema (e.g., extracting pricing, features, or competitor data). Use the n8n Notion node to automatically insert this JSON data as structured rows in a Notion database.

System Prompts & Configuration

The secret to avoiding the "AI tone" is in your system prompts. Always include parameter constraints. If you are building tools or formatting code, strictly define the output format.

Task	Recommended Prompt Architecture
Email Reply	"Adopt a direct, professional tone. No fluff, no 'I hope this finds you well.' Use clear line breaks. Limit to 3 sentences max."
Code Generation	"Output ONLY valid raw code. Do not wrap in markdown tags. Do not explain the code. Use functional components."
Data Extraction	"Extract all numerical data and product features. Output strictly as a minified JSON object matching this schema: { 'price': number, 'features': [] }."

Maintenance and Reality Check

Building your own stack means you are the admin. You will occasionally need to refresh Google OAuth tokens (usually every 7 days in testing environments) or update CSS selectors if a website changes its DOM structure. However, the trade-off is absolute data privacy, zero vendor lock-in, and a system that perfectly maps to your exact technical specifications.