Build Your Own No-Code Automation Stack with Open Source Tools

This playbook shows you how to create powerful, self-hosted automation workflows using only open-source tools — no more expensive Zapier or n8n subscriptions. You'll learn exactly how to integrate scraping, local LLMs, and API agents into a single, cost-effective system.

---

1. Understand the Core Components of Your Stack

Before building anything, know what each tool does in your ecosystem:
- Scrapy, Trafilatura, and Crawlee are for web scraping
- Automatisch powers workflow automation
- MOO Resources helps organize and manage your assets
- Local LLM runners like those in Fastio's 2025 benchmarks let you run models locally

2. Pick Your Scraping Tools Based on Use Case

Choose from:
- Scrapy for robust, scalable scraping
- Trafilatura for fast text extraction
- ScrapeGraphAI for graph-based data extraction
- Crawlee for high-performance crawling and scraping

3. Set Up a Local LLM Runner for Agents

Use tools from Fastio's list to select the best local runner:
- Ollama
- LM Studio
- LocalAI

These enable running large language models without cloud dependency.

4. Integrate with MCP Server Using Open Source Agents

Use the open-source API agent from Agoda to integrate with MCP servers and connect your tools seamlessly.

5. Automate Workflows With Automatisch

Install Automatisch using its AGPLv3 core (commercial use is allowed when not modified). Configure it as your central orchestrator that ties together scraping, LLMs, and APIs.

6. Create Reusable Scraping Prompts Using Scrapy

Write scrapers using Scrapy's flexible Python-based syntax. Use the BSD-3-Clause licensed framework to build custom scrapers for data collection tasks.

7. Extract Clean Text from Web Pages with Trafilatura

Use Trafilatura, Apache-2.0 or GPLv3 licensed, to extract clean, structured text from any webpage — ideal for content aggregation and summarization.

8. Build Graph-Based Scrapers with ScrapeGraphAI

Leverage MIT-licensed ScrapeGraphAI to create graph-based scrapers that extract structured data like relationships between entities.

9. Use Crawlee for High-Performance Crawling

Crawlee, under MIT license, supports headless browser automation and high-speed crawling for dynamic websites.

10. Organize Resources with MOO Resources (Commercial Consideration)

While MOO Resources is not open source, it can help manage your stack assets and tools in a centralized way — note that commercial usage may be restricted.

11. Select a Reliable GNU-Based Operating System

Use GNU/Linux systems for hosting all components, as supported by the GNU Support website (CC BY-SA 3.0 licensed documentation).

12. Choose a Lightweight Web Scraping Library for Simple Tasks

For lightweight tasks, consider BeautifulSoup or Selenium — both are commonly used in open-source Python scraping stacks and have permissive licenses.

13. Connect Everything Through a Custom API Agent

Build a custom agent using MCP server integrations to act as the interface between your tools — leveraging Agoda's open-source agent framework.

14. Set Up an Open Source LLM Inference Endpoint

Use local runners like Ollama or LocalAI to host inference endpoints for your agents without relying on external APIs.