WIRING THE RESEARCH PIPELINE

Frontier Survey + Implementation Architecture
For Timmy's Autonomous Deep Research System

The last time you should need to ask Claude for research.

Alexander Whitestone / Rockachopa • March 24, 2026
For triage by Timmy, Kimi, and code agents

Part 1: Frontier Survey — What Ships Today

This section catalogs every tool Timmy can use to build an autonomous research pipeline. Every entry is verified installable as of March 24, 2026. API-key-required vs. fully self-hostable is flagged.

1.1 Autonomous Research Agent Frameworks

The field exploded in early 2026. Five distinct approaches exist. **Key finding: Local Deep Research** (pip install local-deep-research) runs fully local with Ollama + SearXNG, includes a built-in MCP server, achieves ~95% on SimpleQA, and supports 10+ search sources. **GPT-Researcher** supports Ollama natively with FAST_LLM/SMART_LLM/STRATEGIC_LLM mapping directly to Timmy's cascade pattern.

Framework	Install	Ollama?	License	Verdict
Local Deep Research (LearningCircuit)	pip install local-deep-research	YES. MCP server. ~95% SimpleQA.	MIT	■ BEST sovereign option
GPT-Researcher (assafelovic)	pip install gpt-researcher	YES. SMART_LLM=ollama:qwen3:32b	Apache 2.0	■ Most mature architecture
LangChain Local Deep Researcher	pip install langgraph	YES via Ollama or LMStudio	MIT	Good reference implementation
STORM v2 (Stanford)	pip install knowledge-storm	Partial — needs OpenAI-compat	MIT	Best for long-form articles
HuggingFace Open Deep Research	smolagents examples/	YES via any OpenAI-compat	Apache 2.0	Good code-as-action reference
DeepSearcher (Zilliz)	pip install deepsearcher	YES — Ollama supported	MIT	Best for private data + web

1.2 Search Backends: SearXNG Wins for Sovereignty

Provider	Setup	Cost	Self-Hosted?	Verdict
SearXNG	docker run -d -p 8080:8080 searxng/searxng:latest	$0 forever. No API key.	■ YES	■ RECOMMENDED
SearXNG + Tavily Adapter	docker-compose from le0-star/searxng-docker-tavily-adapter	$0. Drop-in Tavily replacement.	■ YES	Enables Tavily-compat tools free
Tavily	pip install tavily-python + API key from tavily.com	Free: 1K/mo Basic: $30/mo	■ Cloud only	Best AI-optimized results quality
Brave Search	API key from brave.com	$5 free credit/mo ~1,000 queries	■ Cloud only	Cheapest paid alt
DuckDuckGo	pip install duckduckgo-search	$0. No key.	■ (scrapes DDG)	Zero-config fallback

Critical: searxng-docker-tavily-adapter provides a Tavily-compatible REST API wrapping SearXNG. Any tool supporting Tavily (GPT-Researcher, LangChain) can use SearXNG as a drop-in with zero code changes. Fully sovereign. Zero API keys.

1.3 Web Content Extraction: Crawl4AI Already in Agno

Tool	Install	Key Feature	License	Verdict
Crawl4AI v0.8.5	pip install crawl4ai	Anti-bot, Shadow DOM, BM25 filter, LLM-ready md	Apache 2.0	■ Already in Agno! from agno.tools.crawl4ai
trafilatura v2.0	pip install trafilatura	Clean extraction, tables, 60+ langs. No browser.	Apache 2.0	Best for simple pages. ~50ms. Lightweight.
Jina Reader	r.jina.ai/{url}	Instant LLM-ready md. No install needed.	Proprietary API	Emergency fallback

Pattern: try trafilatura first (~50ms, pure HTTP). Fall back to Crawl4AI for JS-heavy pages (~2-5s, uses Playwright browser).

1.4 Vector Stores and Embeddings

Component	Install	Key Feature	Verdict
LanceDB v0.30	pip install lancedb	Embedded, serverless, Rust core. Native hybrid search (vector+FTS+rerank).	■ RECOMMENDED vector store
sqlite-vec v0.1.6	pip install sqlite-vec	SQLite extension. 3 lines to use. Pure C, no deps.	Lightest option for MVP
Qwen3-Embedding 0.6B (NEW — March 2026)	ollama pull qwen3-embedding:0.6b	MTEB top open model. Configurable dimensions 256-8192. 32K context.	■ RECOMMENDED embedding. Replaces nomic-embed-text.
nomic-embed-text v1.5	ollama pull nomic-embed-text	768 dims fixed. 8K context. 274 MB. Previous best.	Still excellent. Proven stable.

1.5 Self-Improving Agent Patterns

Pattern	Source	Core Idea	Relevance to Timmy
Voyager Skill Library	MineDojo (NeurIPS 23)	LLM writes code skills, stores by embedding, retrieves for reuse	DIRECT — crystallization loop for game decisions
ExpeL	LeapLabTHU (AAAI 24)	Extract insights from experience trajectories as NL rules	DIRECT — extract rules from research sessions
Reflexion	noahshinn (NeurIPS 23)	Reflect on failure, generate verbal reinforcement, retry	Research retry pattern when synthesis fails
Hermes Agent Skills	NousResearch	SKILL.md files with YAML. Auto-discovered. 26+ platforms.	DIRECT — already in falsework protocol
Lesson Loop	Multiple (2025-2026)	Mistake → rule → store → load into future context	The sovereignty loop applied to development

Part 2: Implementation Architecture

2.1 System Data Flow

```
User / Dashboard / Gitea Issue ■ ▼ ResearchOrchestrator (src/timmy/research.py) ■ ■■■ 0. Check LanceDB cache (embedding similarity) ■ ■■■ HIT? Return cached result. $0. Done. ■ ■■■ 1. Load template from skills/research/{name}.md ■■■ 2. Generate queries via cascade (Groq → Ollama) ■■■ 3. Search via SearXNG (self-hosted, port 8080) ■■■ 4. Fetch pages: trafilatura fast → Crawl4AI fallback ■■■ 5. Synthesize report via cascade (Groq → Ollama) ■■■ 6. CRYSTALLIZE: embed + store in LanceDB ■■■ 7. Write report to repo (research/{topic}.md) ■■■ 8. File Gitea issues from actionable findings Next time same topic: Step 0 hits. Steps 1-8 skipped.
```

2.2 Component Inventory

File	Action	Purpose	Dependencies
src/timmy/research.py	CREATE	ResearchOrchestrator class	cascade, lancedb, crawl4ai
src/timmy/tools.py	MODIFY	Add web_fetch tool	trafilatura, crawl4ai
skills/research/*.md (6 templates)	CREATE	Research prompt templates	None — markdown files
src/timmy/research_memory.py	CREATE	LanceDB wrapper + embeddings	lancedb, ollama
src/timmy/research_metrics.py	CREATE	Sovereignty metrics tracking	sqlite3 (stdlib)
src/integrations/gitea_client.py	CREATE	Gitea REST API for issues	requests
src/integrations/paperclip/task_runner.py	MODIFY	Add 'research' task type	Existing

2.3 ResearchOrchestrator (Core Implementation)

```
class ResearchOrchestrator: def __init__(self, cascade, searxng_url="http://localhost:8080", db_path="./research.lancedb"): self.cascade = cascade # existing LLM router self.searxng = searxng_url self.db = lancedb.connect(db_path) self._ensure_table() async def run(self, topic, template="state_of_art", **ctx): # STEP 0: CACHE CHECK (sovereignty-first) cached = self._check_cache(topic) if cached: metrics.record("research_cache_hit") return ResearchResult(source="cache", content=cached) # STEP 1: Load template + fill slots meta, prompt = self._load_template(template, topic=topic, **ctx) # STEP 2: Generate search queries queries = await self.cascade.generate( f"Generate 8-12 search queries for: {topic}", format=QueryList.model_json_schema()) # STEP 3: Search via SearXNG snippets = [] for q in queries: r = httpx.get(f"{self.searxng}/search", params={"q":q,"format":"json"}) snippets += r.json().get("results",[])[:5] # STEP 4: Fetch pages (trafilatura fast, Crawl4AI fallback) pages = await self._fetch_ranked(snippets, topic, max=10) # STEP 5: Synthesize via cascade report = await self.cascade.generate( system=prompt, user="Sources:\n" + self._format_pages(pages)) # STEP 6: CRYSTALLIZE into LanceDB vec = ollama_embed("qwen3-embedding:0.6b", topic+" "+report[:500]) self.db.open_table("research").add([{ "topic":topic, "content":report, "embedding":vec, "timestamp":time.time()}]) # STEP 7+8: Write report + file issues path = self._write_to_repo(topic, report) issues = await self._file_gitea_issues(report) metrics.record("research_api_call", cost=self.cascade.last_cost) return ResearchResult(source="web", content=report)
```

2.4 Research Template Format

```
# skills/research/state_of_art.md --- name: state_of_art queries: 8-12 cascade_tier: groq_preferred --- Survey the current state of: {topic} Context: {project_context} Stack: {current_stack} Date: {date} For each tool found: name, install command, version, license. Status: READY | EXPERIMENTAL | DEAD. Output as markdown with tables. Flag API-key vs self-hostable.
```

Six templates: state_of_art.md, tool_evaluation.md, architecture_spike.md, game_analysis.md, integration_guide.md, competitive_scan.md

2.5 Gitea MCP Server (Official)

Gitea ships an official MCP server at gitea.com/gitea/gitea-mcp. Install: `go install gitea.com/gitea/gitea-mcp@latest`. This gives Timmy's Agno agent read/write access to issues, PRs, repos via MCP — closing the MCP gap identified in the project feedback doc. For Python-only, use the REST API: POST /api/v1/repos/{owner}/{repo}/issues with token auth.

Part 3: Sovereignty Metrics + Build Order

Metric	Week 1	Month 1	Month 3	Graduation
Cache hit rate	10%	40%	80%	>90%
API cost/task	$1.50	$0.50	$0.10	<$0.01
Time to report	30 min (auto)	15 min	5 min	<1 min
Human involvement	Review (10 min)	Review (5 min)	Approve (1 min)	None

Build Order (5 Days Total)

Day	Task	Install Commands
1	Deploy SearXNG + install LanceDB + wire Crawl4aiTools	docker run -d -p 8080:8080 searxng/searxng pip install lancedb trafilatura httpx ollama pull qwen3-embedding:0.6b
2	Create 6 research templates + start ResearchOrchestrator	mkdir -p skills/research/ (write 6 .md template files)
3	Finish ResearchOrchestrator + test end-to-end pipeline	New: src/timmy/research.py Test: python -m timmy.cli research 'test topic'
4	Gitea issue creation + trafilatura fast-path	Use Gitea REST API or gitea-mcp Add trafilatura fallback in web_fetch
5	Paperclip task integration + sovereignty metrics	Modify task_runner.py + new research_metrics.py Import all existing PDFs into LanceDB

After day 5, every research question Timmy answers gets indexed. Every indexed answer makes the next question cheaper. The compound interest of crystallized knowledge.

The last time you should need to ask Claude for research.

Timmy Time Project • Research Pipeline Spec • v1.0