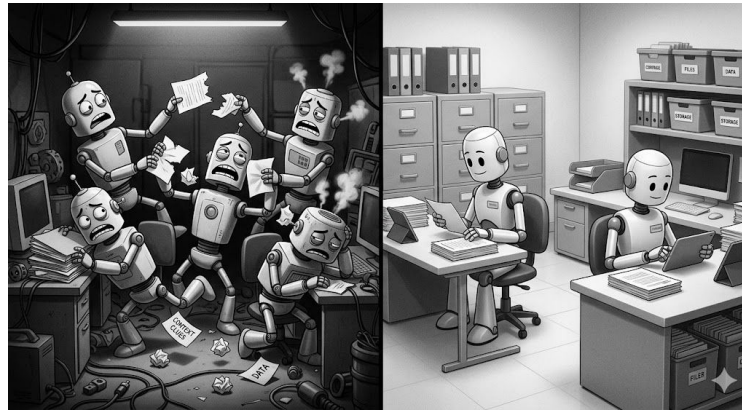


Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

— LinkedIn 2026



Generate a linkedin post related to AI engineering our deep research workshop

Writing



Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.



Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

- Recognizable “slop” words (“—”!!!)

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

Most teams don't lack effort.

' words ("—"!!!)

They lack ownership.

g persons **Most People Track Their Jobs — Not Their Growth**

Most teams answer 'best model' because it's easier to evaluate. But



Louis-François Bouchard  • You

Making AI accessible. 🎥 What's AI on YouTube. Co-founder at T...

[Visit my website](#)

2mo • 

Most AI projects fail before you write a single line of code.

Not because the tech is hard.

Because you skipped the questions that actually matter.

Three tools. Three different jobs.

Most teams are paying for all of them without knowing which does what.

M
AI
THE
NOT

- Recognizable “slop” words (“—”!!!)
- Too general, nothing personal

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

- Recognizable “slop” words (“—”!!!)
- Too general, nothing personal
- Outdated/hallucination

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

- Recognizable “slop” words (“—”!!!)
- Too general, nothing personal
- Outdated/hallucination
- Typical “AI phrases”

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

- Recognizable “slop” words (“—”!!!)
- Too general, nothing personal
- Outdated/hallucination
- Typical “AI phrases”
- Meaningless / shallow

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

- Recognizable “slop” words (“—”!!!)
- Too general, nothing personal
- Outdated/hallucination
- Typical “AI phrases”
- Meaningless / shallow

Lets delve deep into the intricacies of Deep Research and Agentic AI most companies miss.

It's not about what but how you leverage today's rapidly evolving landscape of the best technologies like GPT-4 and Llama 3.

Solution: Add proper research + proper writing

Context

We create AI engineering courses and tons of videos, articles and other technical content.

Technical writing == AI engineers + writer + editor + (a lot of) time == \$\$\$\$

Can we automate this?

-> We did

... well, a (large) part of it.

Context

We built a multi-agent pipeline replacing the research + technical writing processes.

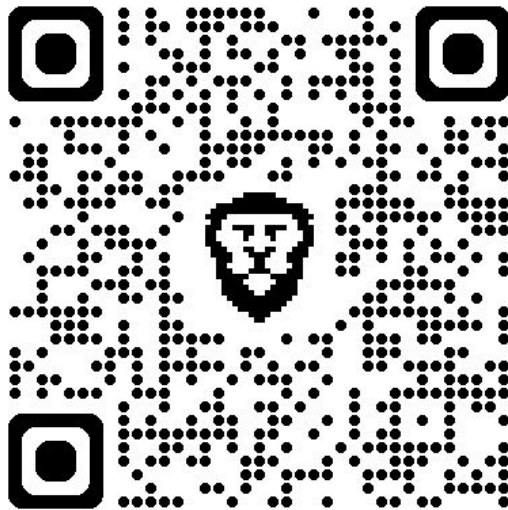
Topic in -> deep research agent -> technical article (with code, images and non-slop writing)

And used it to build a (40h+) course, teaching how to build and deploy it with best practices.

In this (2h) workshop, we share a “compact” version of the deep research + writing system we built (and use) for shorter content (LinkedIn-type posts) , and what we learnt building it...

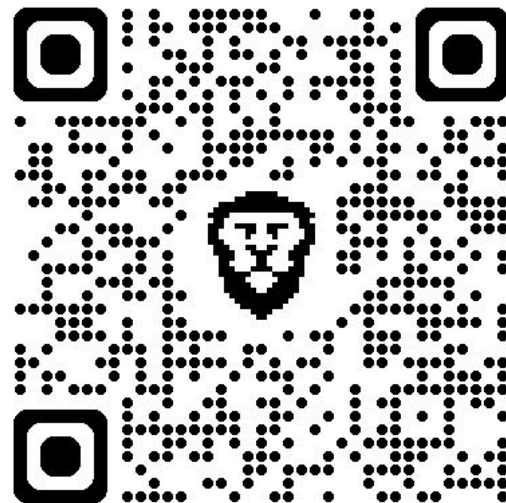
Open Up the GitHub Repository

<https://github.com/iusztinpaul/designing-real-world-ai-agents-workshop/>



Today's Plan

1. **Building AI software**
 - a. **Workflows, agents, tools...**
 - i. Definitions, tips and bringing everyone to the same level
 - b. **(our) Deep Research System as an example**
 - i. How we tackle agentic AI projects
2. **Building the Deep Research Agent**
3. **Building the LinkedIn Writing Workflow (That Passes Slop Detectors)**
4. **Adding Observability: Monitoring & Evaluation**





Louis-François Bouchard, Co-founder & CTO @ Towards AI

- Ex-PhD student at Mila, Montréal
- Educator in AI since 2019
- AI R&D and AI products development since 2020

Samridhi Vaid, ML/AI Engineer

- M.Sc. Computer Science, University of Alberta
- Technical writer & consulting since 2024



Paul Iusztin, Founder @ Decoding AI

- Building AI software for 8+ years
- Teaching AI for 4+ years
- Author of the LLM Engineer's Handbook bestseller

The AI Engineering Problem Space

Real-world constraints govern every decision:

- Cost per task (can swing a lot based on architecture decisions).
- Latency requirements (real-time vs. batch).
- Quality bar (acceptable error rate).
- Compliance & data privacy.

The AI Engineering Stack

- Context (RAG/CAG + memory) • Tools • Orchestration • Reasoning • Evaluation

Capability options (“autonomy slider”):

Simple Prompts → Workflows → Single Agent → Multi-Agent Systems



More Control, lower costs

More Autonomy, higher costs

Many products stop at workflows; agents are optional and add risk

The foundations of workflows and agents



The foundations of workflows and agents

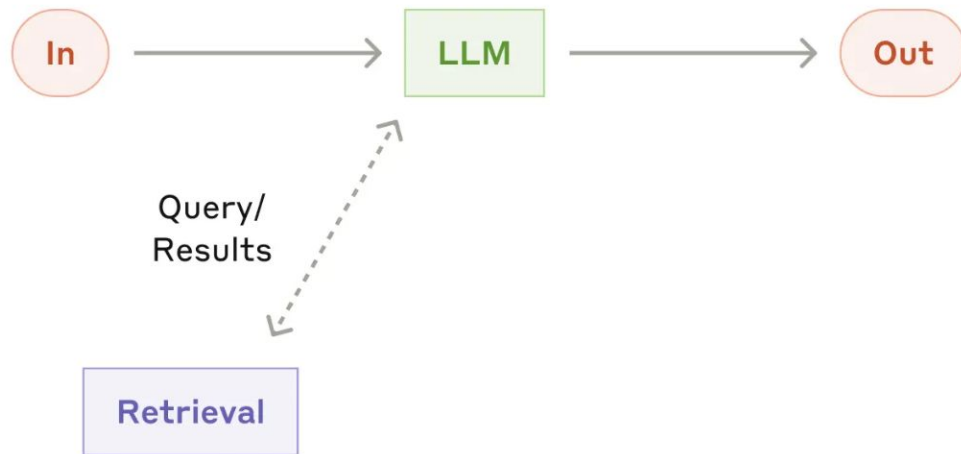
An LLM... but augmented



The foundations of workflows and agents

An **LLM**... but **augmented**

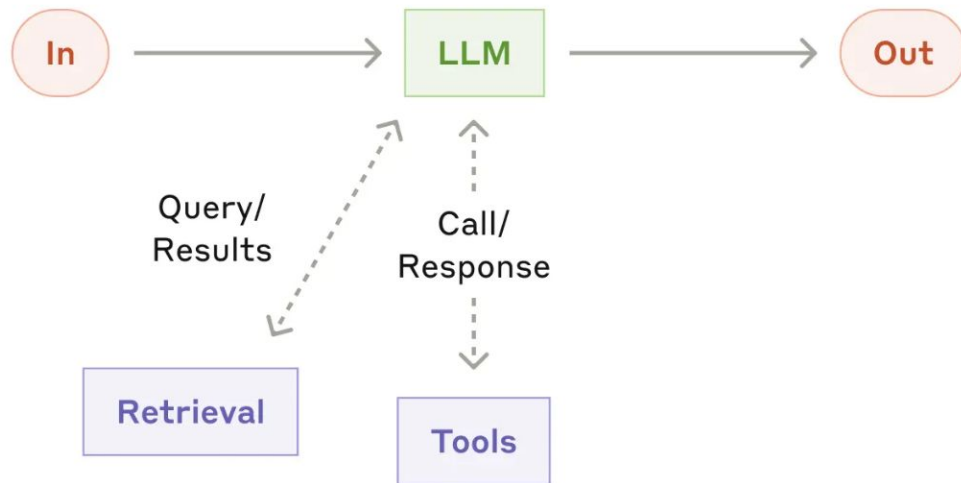
+ Additional data



The foundations of workflows and agents

An **LLM**... but **augmented**

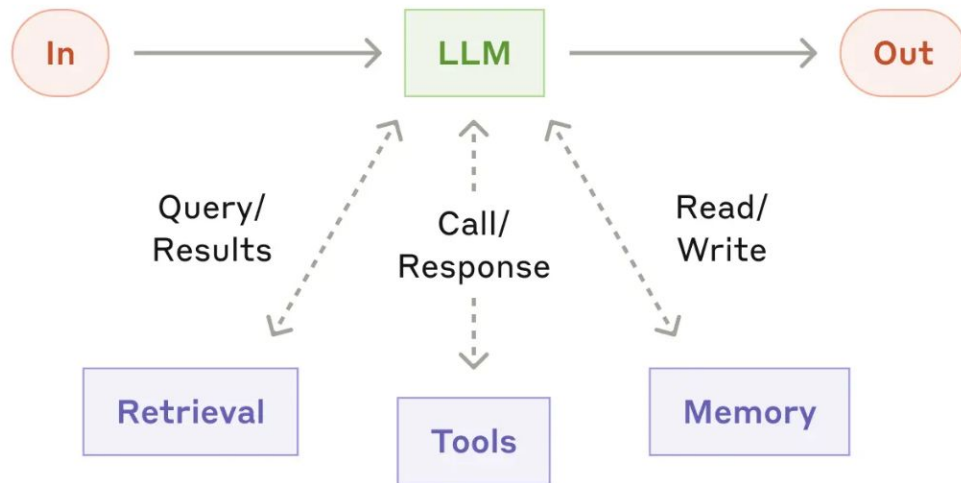
- + Additional data
- + Tools



The foundations of workflows and agents

An **LLM**... but **augmented**

- + **Additional data**
- + **Tools**
- + **Memory**

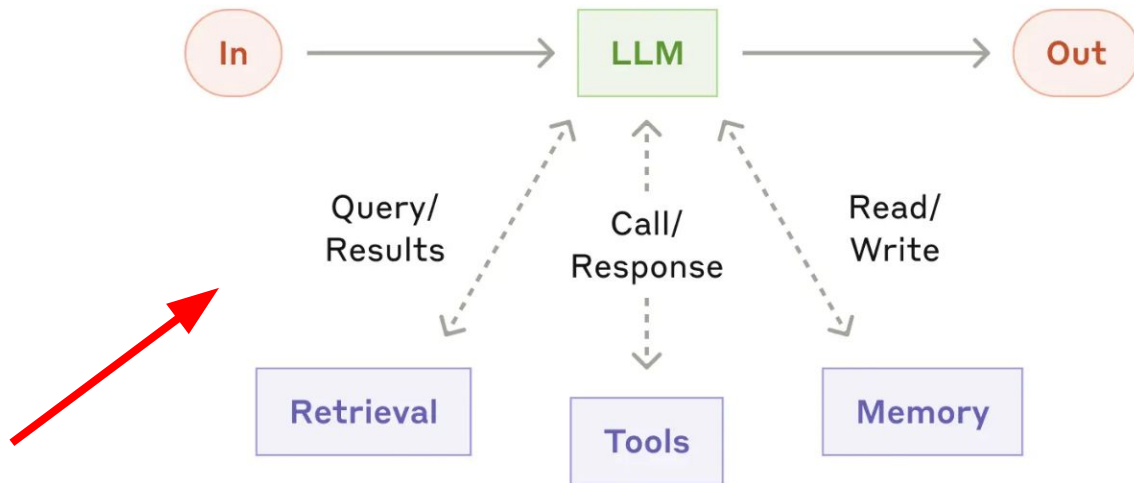


The foundations of workflows and agents

An **LLM**... but **augmented**

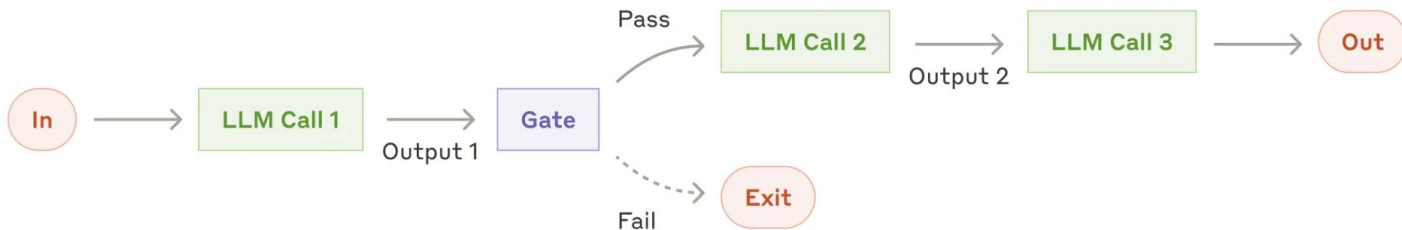
- + **Additional data**
- + **Tools**
- + **Memory**

Not an agent!



The foundations of workflows and agents

Useful when a task can be easily broken down into simple cases (“prompt chaining”)

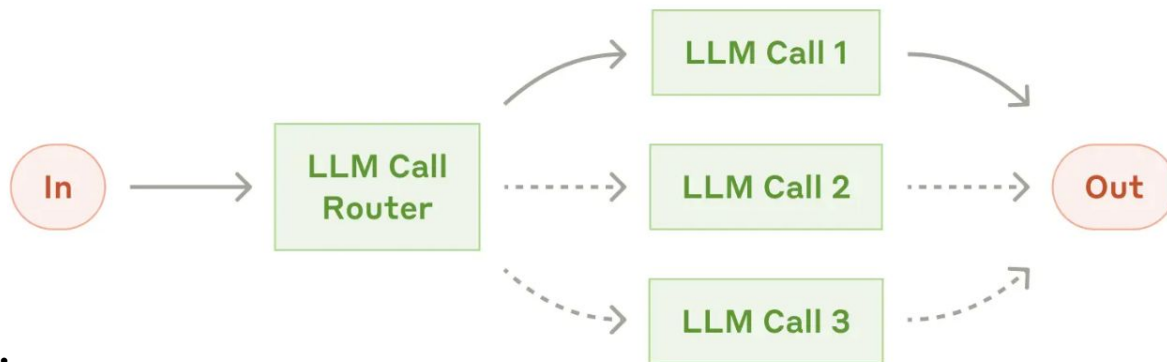


We reduce the latency and complexity of a task!

The foundations of **workflows** and agents

We can choose what to do based on the case: router (or “orchestrator”)!

-> useful when it can be determined in advance!



We might want to:

- Use a smaller model
- Classify a result (with or without an LLM)
- Perform intermediate tasks (without LLMs)
- Use a fine-tuned model for a specific case
- etc.

The foundations of workflows and agents

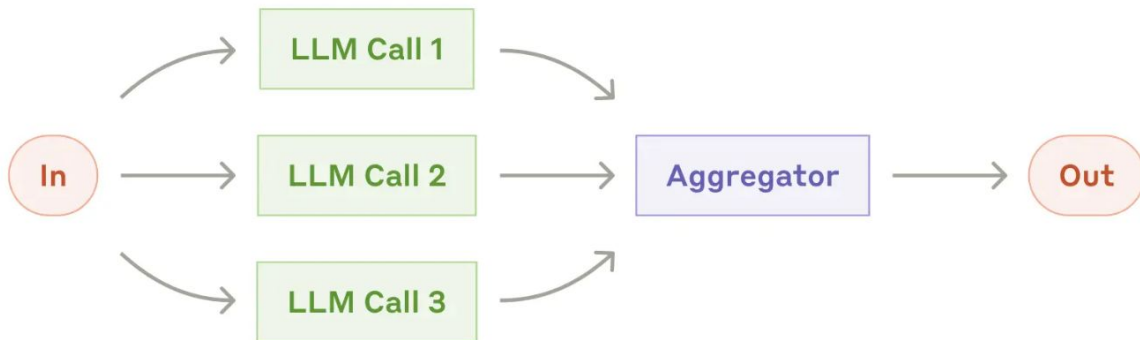
An addition would be to do it in parallel!

-> to be more efficient

Parallelization

-> to improve results

Majority vote

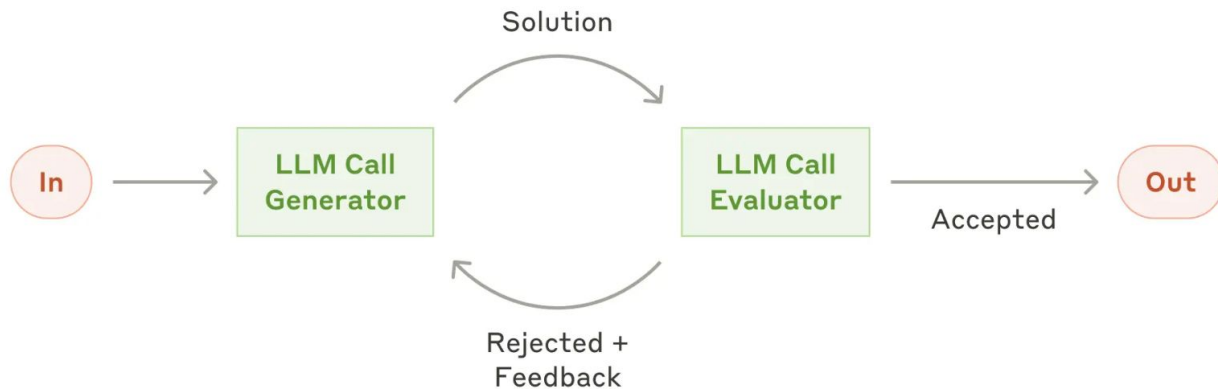


The foundations of workflows and agents

We can even (especially) add loops!

Generator + evaluator

*Useful when we can **evaluate** what we want + **measurable** outputs + (human) **feedback** improves the LLM!*



Some call this agents or “LLMs in a loop”

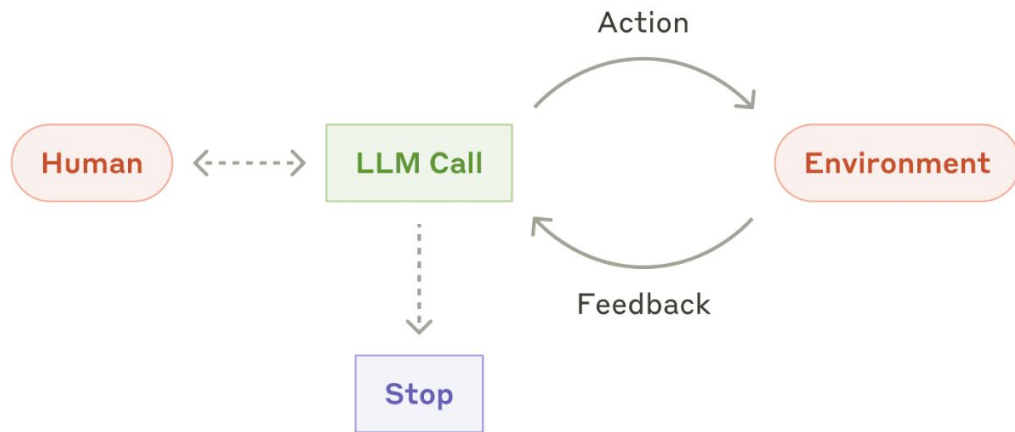
The foundations of workflows and agents

... so, when does it become an agent (or several agents)?!



The foundations of workflows and **agents**

... so, when does it become an agent (or several agents)?!



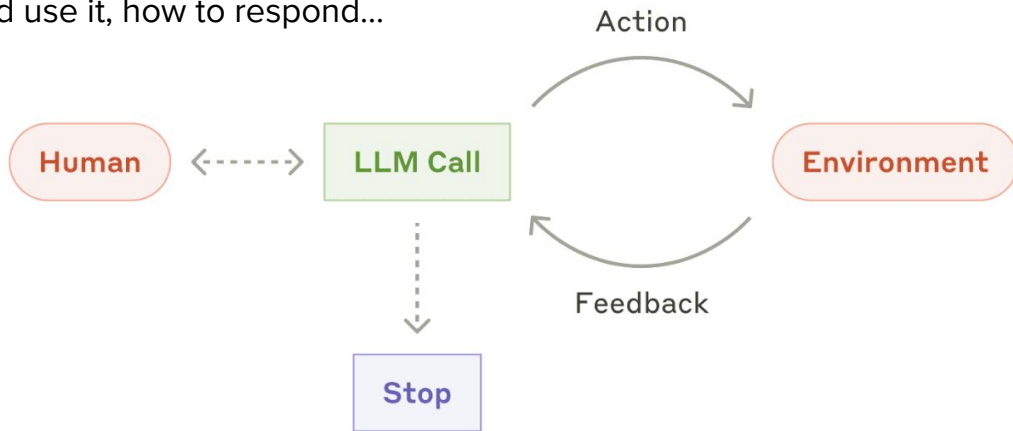
The foundations of workflows and agents

... so, when does it become an agent (or several agents)?!

Short answer:

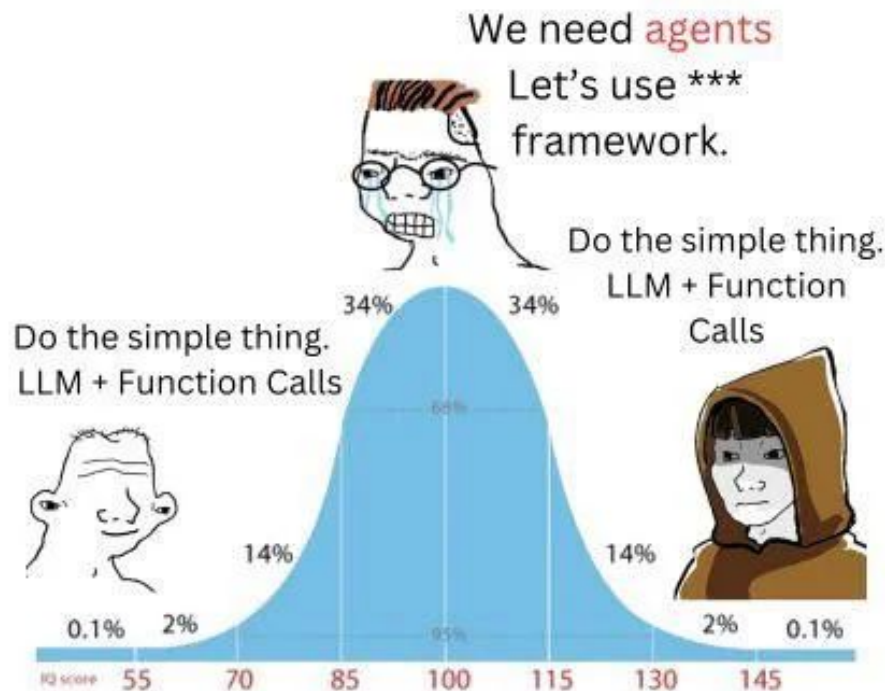
It takes autonomous actions and reacts to what happens according to the environment

- Plans, decides which tool to use and if it should use it, how to respond...



When to use (and not use) agents

Always use the simplest possible solution!



When to use (and not use) agents

Always use the simplest possible solution!

Simple Prompts



Workflows



Single Agent



Multi-Agent Systems



More Control, lower costs

More Autonomy, higher costs

Q1: Does the model already know enough?

Few-shots + system prompt

-> Prompting

When to use (and not use) agents

Always use the simplest possible solution!

Simple Prompts



Workflows



Single Agent



Multi-Agent Systems



More Control, lower costs

More Autonomy, higher costs

Q2: Do you need external context?

If context is known at/before query time and <200K tokens

-> **CAG/long context**

When to use (and not use) agents

Always use the simplest possible solution!

Simple Prompts



Workflows



Single Agent



Multi-Agent Systems



More Control, lower costs

More Autonomy, higher costs

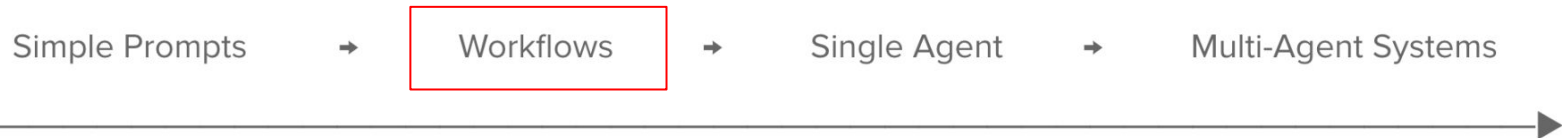
Q3: Is context unknown until query time?

Private? Too recent? Domain-specific? Citations required?

-> RAG (inject knowledge)

When to use (and not use) agents

Always use the simplest possible solution!



More Control, lower costs

More Autonomy, higher costs

Q4: Are there multiple steps in a fixed sequence?

Order you can pre-determine? Various steps in sequence?

-> **Workflow** (orchestrate LLM calls + tools + logic with code)

A Workflow in Practice

Example: Support ticket handling system

1. Receive ticket
2. Classify
3. Route to team
4. Draft response
5. Validate against policy
6. Send

Key point:

We can list each step as they come.

Each step may use an LLM, but the **order never changes**.

Building this as an agent adds overhead without adding capability.

When to use (and not use) agents

Always use the simplest possible solution!

Simple Prompts



Workflows



Single Agent



Multi-Agent Systems



More Control, lower costs

More Autonomy, higher costs

Q5: Must the system take actions or branch dynamically?

API calls, DB writes, multi-step reasoning...

-> Agent + tools

An Agent in Practice

Real **example**: AI-assisted marketing content generation chatbot for a CRM platform

- **Initial design (from client): many specialized agents**
 - Orchestrator agent
 - A few content generation agents
 - Validation agent
 - Optimization agent
 - Security agent
 - Scoring agent
 - Normalization agent

On paper: sounds cool

In reality: unnecessary coordination overhead, duplicated code, tons of complexity...

An Agent in Practice

Real **example**: AI-assisted marketing content generation chatbot for a CRM platform

Actual flow is always:

Plan (agent 1) → **Retrieve** (tool) → **Generate** (agent 1 with different prompts) → **Validate** (fixed loop with conditions) → **Fix** (agent 1)

The tasks were:

- Sequential
- Tightly coupled (always similar! **Goal** = Generate marketing content for x,y,z)
- Context-dependent

Splitting decisions across agents caused:

- Information silos (Lost global context between agents)
- Potential handoff errors

An Agent in Practice

Tools can be specialized without adding agents (they are capabilities, not intelligence!):

Tools can have:

- Their own system prompts
- Their own validation logic
- Even their own LLMs

From our real example:

- **Validation tool** with strict error detection
- **Personalization tool** with schema enforcement
- **SMS tool** with deterministic character limits and writing style

Result:

Tools are specialists, but one **shared decision maker** with a shared **global context**.

The Constraint: The Context Window

An agent's context is made of:

- **System (and user)** instructions (persona, constraints)
- **Tool** definitions (schemas)
- Few-shot **examples** (teaching by examples)
- Retrieved **data** (RAG)
- Entire **conversation history** (thought → action → observation loops)

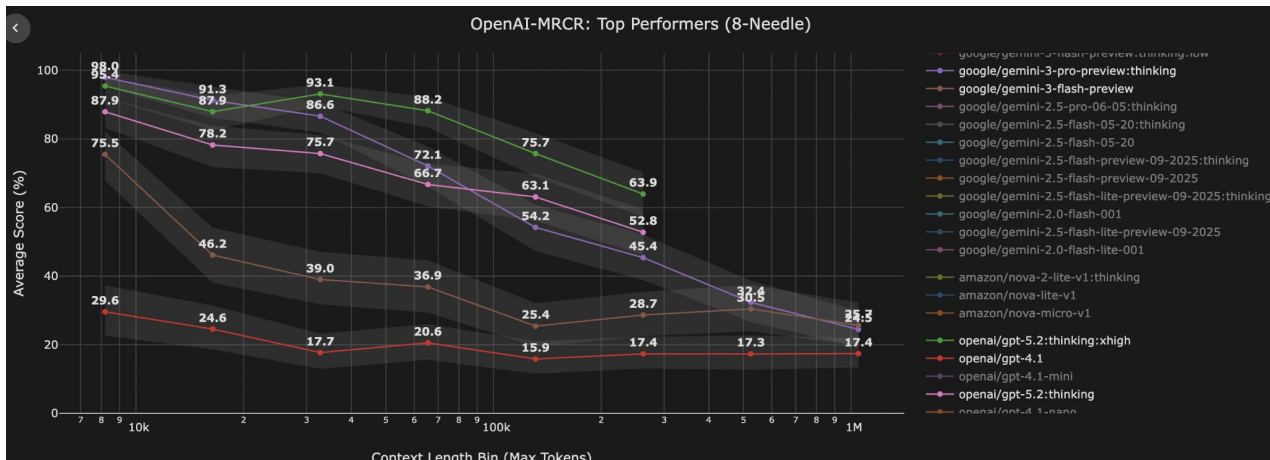
The Constraint: The Context Window

An agent's context is made of:

- **System (and user)** instructions (persona, constraints)
- **Tool** definitions (schemas)
- Few-shot **examples** (teaching by examples)
- Retrieved **data** (RAG)
- Entire **conversation history** (thought → action → observation loops)

The problem: context grows at each step, and performance degrades:

- **i.e. Context Rot:** Performance degrades as context grows (even before hitting limits)
- **Noisy Context:** Irrelevant information distracts the LLM
- **Lost in the Middle:** Information in the middle of long prompts gets ignored



Managing the Context Budget

Rule: Keep context lean and relevant (we must engineer the context)

1. Trimming

- a. How: Sliding window: keep last N turns, discard older
- b. Risk: May discard critical early instructions

2. Summarization

- a. How: Condense old interactions via LLM
- b. Risk: May lose crucial details; adds latency/cost

3. Selective Retrieval

- a. How: Reconstruct context per step using embeddings to find most relevant pieces
- b. Risk: May fail to retrieve the relevant context and lose some; adds latency/cost

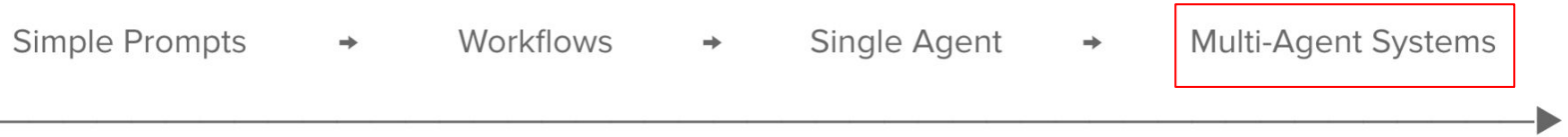
4. ... and many others (see Claude Code Leaked compaction methods!)

5. Delegate to external tools AND/OR “sub-agents” with each their own context

(as in Codex, claude code, claws...)

When to use (and not use) agents

Always use the simplest possible solution!



More Control, lower costs

More Autonomy, higher costs

Q6: Do you need multiple specialized agents?

>20 tools, different isolated reasoning, autonomous decision making, tool variability, security compliance, budget...

-> Multi-agent

Why am I talking about all this?

Because AI products are never just agents, just a simple workflows or just LLM calls and tools.

They combine all of them.

AI Engineers need to understand how to build such complex systems.

And **deep research** systems are a very good complex example

-> they integrate and combine all of these techniques

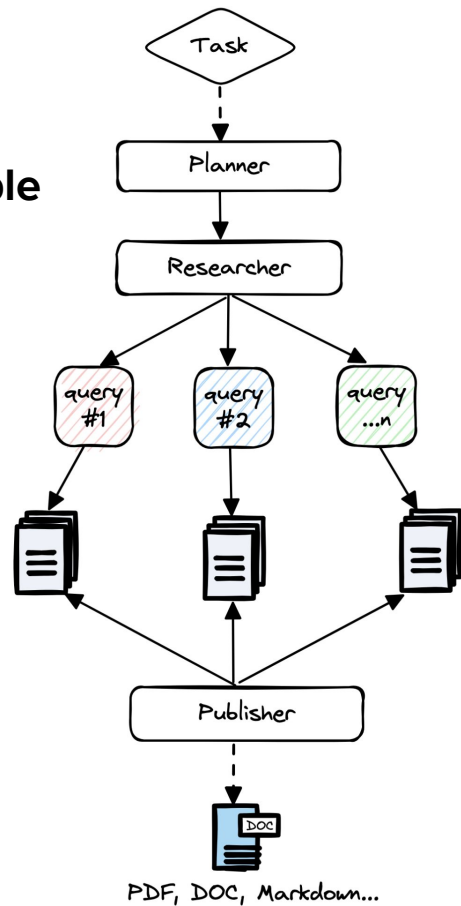
And before we start building our own, let's cover what it is...

Deep Research Systems

Reasoning model + planning + autonomy + tools + internet + reliable (grounded) + feedback loops

=

Deep research



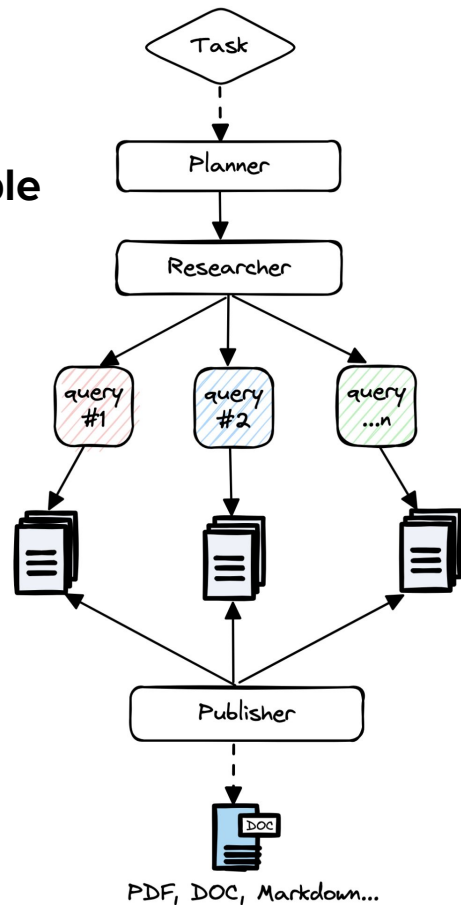
Deep Research Systems

Reasoning model + planning + autonomy + tools + internet + reliable (grounded) + feedback loops

=

Deep research

1. **Tools:** search, browsing, document/video analysis, note-taking
2. **Environnement:** web + user-provided sources + APIs/knowledge bases
3. **Objective:** “goal-directed research loop”



Deep Research Systems

Reasoning model + planning + autonomy + tools + internet + reliable (grounded) + feedback loops

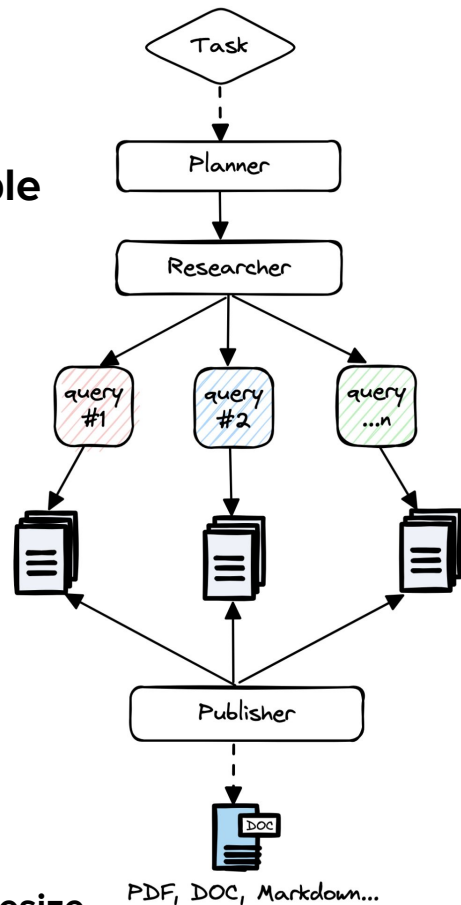
=

Deep research

1. **Tools:** search, browsing, document/video analysis, note-taking
2. **Environnement:** web + user-provided sources + APIs/knowledge bases
3. **Objective:** “goal-directed research loop”

More than a chatbot:

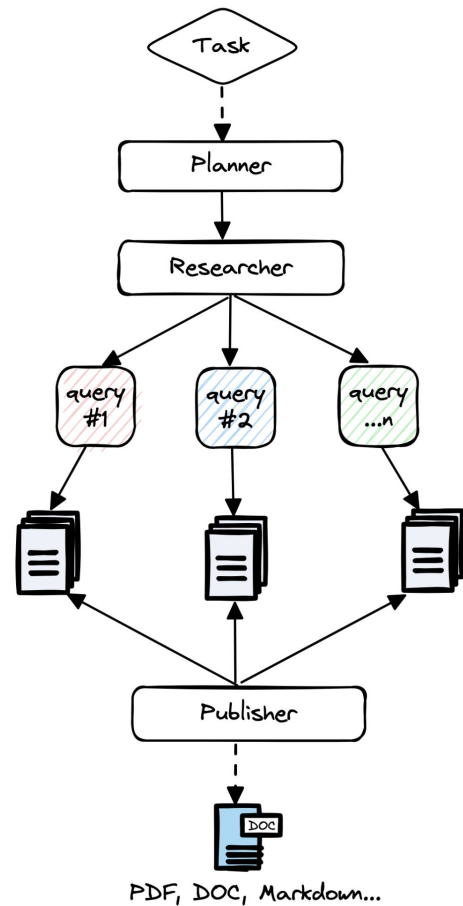
An autonomous research assistant that can **plan**, **search**, **inspect**, **pivot**, and **synthesize**.



Deep Research Systems

Perfect project to learn to build end-to-end systems!

so we built our own (to teach how to build it)



Multi-Agent System for Research + Article Writing

- **Project:** End-to-end technical article production system from a topic.
- **Goal:** Take a **topic**, research it deeply, **write** a publish-ready technical article/course lesson.
- **Challenge:** **precision/recall** in research: finding as much relevant information as possible while finding as little irrelevant information as possible, **hallucinations**, "**AI Slop**", **incorporating** (a lot of) **human feedback** & more in writing.

Multi-Agent System for Research + Article Writing

Question 0: Is this actually worthwhile?

- **Analysis:**
 - High-quality **technical content is expensive to produce.**
 - Research **needs strong precision/recall.**
 - Writing must **avoid hallucinations and AI slop** and be of **high quality.**
 - Current deep research tools are really **exhaustive** and have a lot of **noise.**
- **Decision:**
 - Yes! This is worthwhile as a **writer-augmentation (human-in-the-loop) system** first.
 - Aim to augment AI engineer writers now, and automate larger parts only where quality is reliable.

Multi-Agent System for Research + Article Writing

Question 1: What architecture should we use? Single agent or multi-agent?

- **Analysis:**
 - **Research is exploratory** (find information through various sources) and directed by human feedback: "Search, read, pivot, search again."
 - **Writing is deterministic**, follow tone, structure..., final edits with human feedback (human review at the end only): "Draft, critique, edit, format." (Workflow pattern)
 - **Conflict:** Research needs flexibility; Writing needs constraints.
- **Decision:** Split into two systems:
 - **Research agent** (exploratory, dynamic).
 - **Writer agent** (deterministic, consistent).

Multi-Agent System for Research + Article Writing

Question 2: Use an orchestrator (main agent)? Keep the two agent together or separate?

- **Analysis:**
 - We **pivoted** (a few times!). After testing, users almost always **used both agents once, without alternating between them.**
 - **Iterating with the writer agent itself is enough** (to add new small topics to the article, without the need for research).
 - If a big change is needed, the user would **run the process again from scratch.**
 - **The two agents work on the same artefacts**, they are dependent.
- **Decision:**
 - Two separate agents, **run sequentially** without orchestration between them.
 - Two agents **in the same project** to share context through dedicated files (**this is also useful for AI-assisted coding**).

Multi-Agent System for Research + Article Writing

Question 3: How should the first agent (research agent) **behave**?

- **Behaviour (after several tests...):**
 1. **Read a human-written article guideline** to understand the topic to research.
 - the more we define the agent's goal (less freedom), the better
 2. (optional) **Scrape user provided webpages**, youtube videos, and github repositories to gather more context (super helpful!).
 3. Iteratively do **web searches to learn more** about the topic and gather material, following **user feedback** after each batch of web searches. (user can add URLs here too!)
 4. **Revise all the gathered content** and filter to keep only the most relevant and trustworthy content.
 5. **Write all the gathered content into a final artefact**, which will be used by the writer agent.

Multi-Agent System for Research + Article Writing

Question 4: How to reliably **scrape web content AND do web searches**?

Don't reinvent the wheel + divide into simpler problems

- **1 - Scraping solution:** use **Firecrawl** & [agentic scraping](#)
 - Websites block suspected bots: scraping services can act like users and get to the webpage (also: IP rotation).
 - Websites work with HTML: scraping services can parse the HTML and extract the relevant content, which is not a straightforward task to make work on ALL websites!
 - Many websites are SPAs (single-page applications), content is often rendered dynamically
- **2 - Web searches solution:** use **Perplexity** (or Gemini with grounding)
 - Get **precise answers to search queries with citations** (links to sources -> we can filter the untrustworthy ones).

Multi-Agent System for Research + Article Writing

Question 5: How to manage YouTube videos?

- **Problem:** A lot of great information is on youtube (e.g. [What's AI!!](#) 😊). **We want to pass only text information to the writer agent** to simplify its job. How do we **convert a video to text**?
- **Analysis:**
 - We **tried different models** to transcribe videos to text (models are evolving quickly on this).
 - Usually it's needed to download the video, and then upload it to the LLM model. With one exception: **Gemini handles this directly, just provide the YouTube video URL.**
- **Solution:** We use **Gemini to transcribe the video and describe it**, as it's one of the best models and the simplest to integrate.

Multi-Agent System for Research + Article Writing

Question 6: How to get content from GitHub?

- **Problem:** We want the **user to be able to specify a GitHub URL** and download all the relative files. It should **work on the private repositories of the user** (so, it's not possible to use scraping services for this, we need to use the GitHub APIs with their token).
- **Solution:** Use the **`gitingest` open-source library** which **formats GitHub content into Markdown** format that is easily understandable by LLMs. It's possible to **provide a GitHub token for access** to the user's private repositories.

Multi-Agent System for Research + Article Writing

Question 7: What **library/framework** do we use to develop the research agent?

- **Architecture:** MCP server with FastMCP
 - **One MCP tool for web searches** with Perplexity, **one for scraping** with Firecrawl, **one for converting YouTube** videos to text with Gemini, etc.
 - MCP allows to use the tools also with other MCP clients.
 - For example, it's possible to run the research agent in Cursor/VS Code...!
- **Orchestration (where is the full “recipe” (procedural memory) of how to use the tools in sequence located?):** it's defined in an MCP prompt.
 - **When the MCP client connects to the MCP server**, it can **read the “recipe” of how to do the full research from an MCP prompt** of the MCP server, and then the MCP client will start using all the tools knowing how they work together!

Multi-Agent System for Research + Article Writing

Question 8: How do we **setup** the research agent project?

- Python, most used language for AI projects.
- `uv` for setting up the project and managing dependencies (a better and faster `pip`).
- GitHub for code collaboration, versioning, backup.

```
mcp_server/
├── .env                # Environment variables
├── .python-version     # Python version
├── .mcp.json.sample    # MCP server configuration sample
├── pyproject.toml      # Project configuration (project name, dependencies, etc.)
├── uv.lock             # Dependency lock file
├── src/               # Source code
│   ├── server.py      # Main entry point
│   ├── app/           # Business logic handlers
│   ├── config/        # Configuration (settings, constants, prompts)
│   ├── models/        # Data models (schemas)
│   ├── prompts/       # MCP prompts (one file for each prompt)
│   ├── resources/     # MCP resources (one file for each resource)
│   ├── routers/       # MCP endpoint registration
│   ├── tools/         # MCP tools (one file for each tool)
│   └── utils/         # Shared utilities (file I/O, LLM, logging)
```

Multi-Agent System for Research + Article Writing

Question 9: What **model** to use for each tool and step?

- **Solution:**
 - YouTube video transcription (hard task) -> Gemini 3.0 Pro.
 - Cleaning of scraped webpages (lot of tokens, easy task) -> Gemini 3.0 Flash.
 - Selection of relevant and trustworthy sources (hard task) -> Gemini 3.0 Pro.

Other providers would have been ok (but Gemini is useful to get youtube transcripts from links directly & free tier for our students!).

Using a single provider makes development easier, but higher quality is usually obtained by testing a lot of models from different providers.

Multi-Agent System for Research + Article Writing

Question 10: How does the research agent **communicate with the user**?

- **Solution:** the research agent, following the instructions provided in the MCP prompt, calls tools and **stops to ask the user for feedback whenever instructed**.
 - The MCP prompt says:

```
**Workflow:**
```

```
1. Setup:
```

```
1.1. Explain to the user the numbered steps of the workflow. Be concise. Keep them numbered so that the user can easily refer to them later.
```

```
1.2. Ask the user for the research directory, if not provided. Ask the user if any modification is needed for the workflow (e.g. running from a specific step, or adding user feedback to specific steps).
```

```
1.3 Extract the URLs from the ARTICLE_GUIDELINE_FILE with the "extract_guidelines_urls" tool. This tool reads the ARTICLE_GUIDELINE_FILE and extracts three groups of references from the guidelines:
```

- "github_urls" - all GitHub links;
- "youtube_videos_urls" - all YouTube video links;
- "other_urls" - all remaining HTTP/HTTPS links;
- "local_files" - relative paths to local files mentioned in the guidelines (e.g. "code.py", "src/main.py").

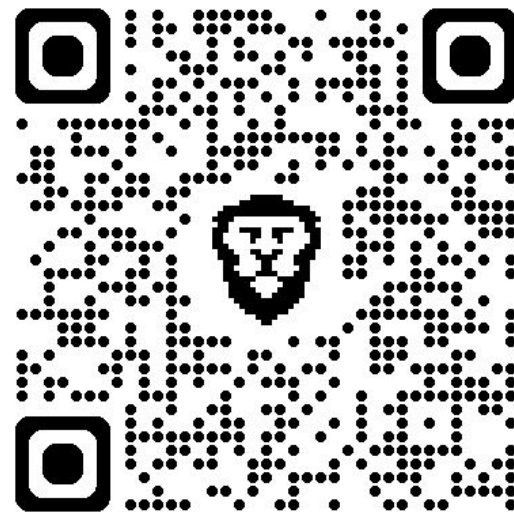
```
Only extensions allowed are: ".py", ".ipynb", and ".md".
```

```
The extracted data is saved to the GUIDELINES_FILENAMES_FILE within the NOVA_FOLDER directory.
```

- All handled through files for easy user edits and transfer between agents and the user.

Part 2: Building a Deep Research Agent

Scan to follow along



<https://github.com/iusztinpaul/designing-real-world-ai-agents-workshop/>

Part 2: Building a Deep Research Agent

What: An agent that researches any topic by searching the web, analyzing YouTube videos, and compiling a cited report

How: MCP (Model Context Protocol), the open standard for giving agents tools and data

Key idea: Agent = the brain (reasoning), MCP Server = the hands (capabilities)

Built with: FastMCP (Python) + Claude Code as the agent runtime

Our MCP Server

Tools = actions the agent can DO

deep_research → Gemini + Google Search → answer + sources

analyze_youtube_video → Gemini native video → transcript

compile_research → reads .memory/ → assembles research.md

Prompt = instructions the agent can FOLLOW

research_workflow → step-by-step guide for the full research flow

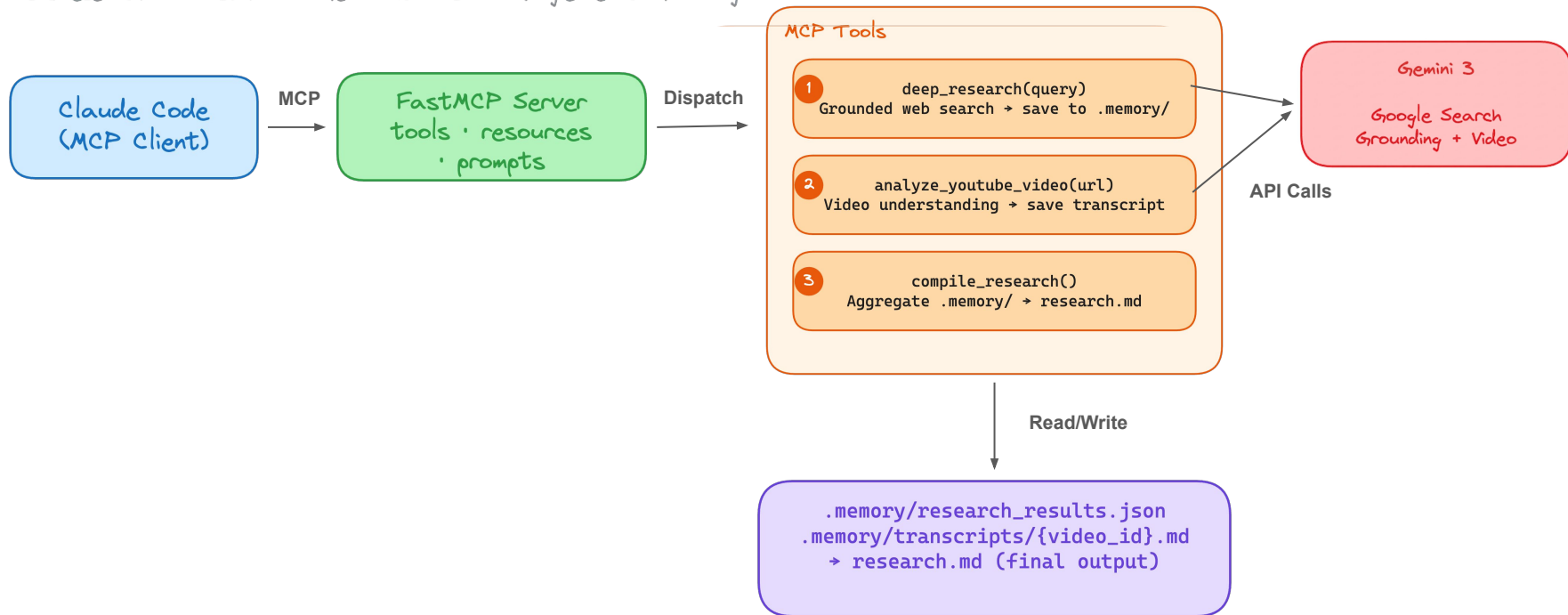
Resource = data the agent can READ

config/research → exposes model names, version, feature flags

Architecture: How It All Connects

Deep Research Agent — MCP Server Flow

src/research/ • FastMCP • Gemini 2.5 Flash • Google Search Grounding



Connecting to Claude Code

.mcp.json in project root → Claude Code auto-connects on startup

```
{  
  "deep-research": {  
    "command": "uv",  
    "args": ["run", "fastmcp", "run", "src/research/server.py"]  
  }  
}
```

Stdio: Claude Code launches server as a child process (stdin/stdout)

Live Demo: Deep Research Agent

- Show `.mcp.json` config
- `/mcp` to verify connection, `/tools` to see registered tools
- Ask Claude Code to research a topic + analyze a YouTube video
- Watch it autonomously call tools, identify gaps, and compile results

Agent Skills

Open, lightweight building blocks that give AI agents new capabilities and workflows.

User request → Skill provides the know-how → Tools do the work

Just a markdown file: `.claude/skills/research/SKILL.md`

name: research # becomes /research

description: “Run deep research on any topic...”

1. Load `research_workflow` MCP prompt
2. Follow workflow using tools
3. Use current directory as `working_dir`

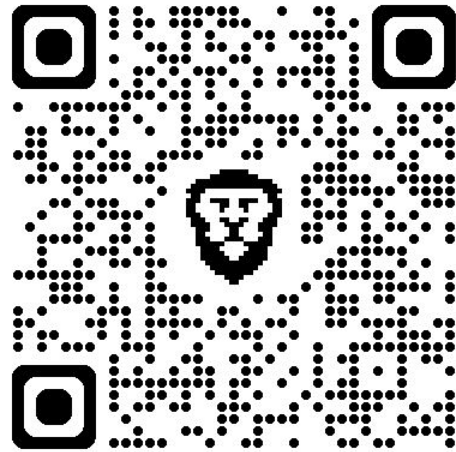
Progressive disclosure: only frontmatter loads at startup, full body on trigger

Our /research Skill: How It All Connects

/research “topic”

- Claude Code matches the skill from frontmatter
- Loads the full SKILL.md on demand
- Follows the research workflow
- Uses tools and references as needed
- Produces a structured research output

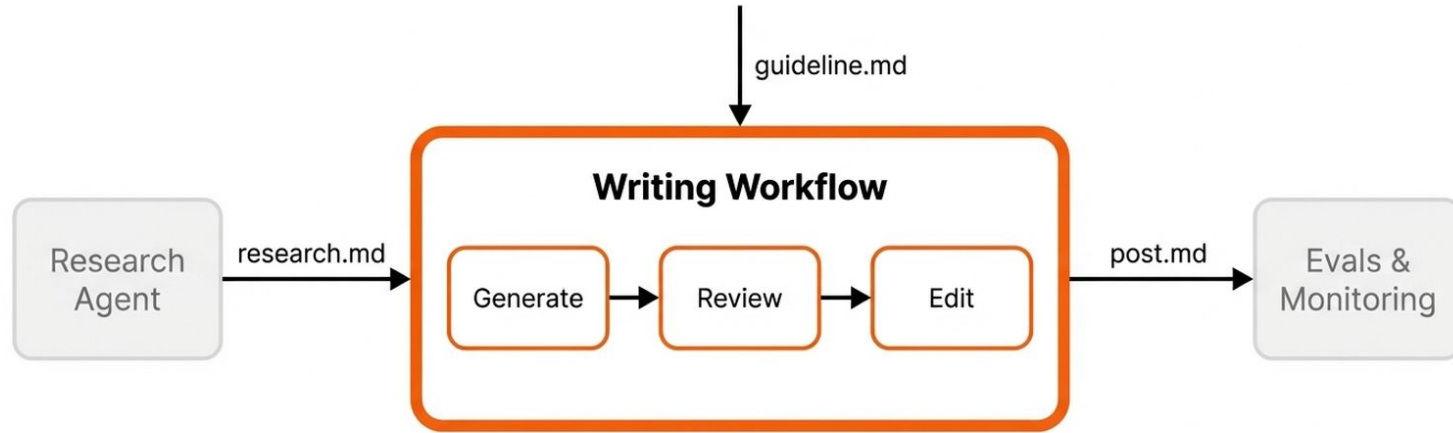
Consistent flow, reusable via Git, discoverable in the command menu



Part 3: The LinkedIn Writing Workflow

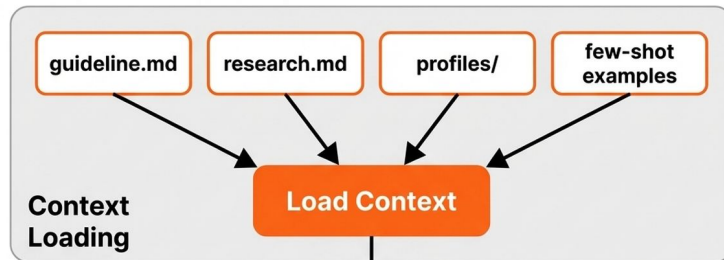
From research to polished posts that pass slop detectors

Understanding the Full System Architecture



Zooming Into the Writing Workflow Architecture

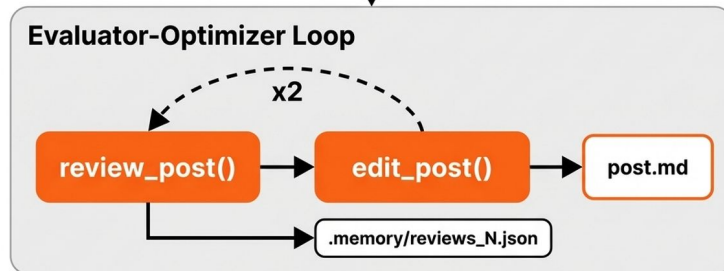
Phase 1: Load the context



Phase 2: Generate initial draft of the post



Phase 3: Improve the post



Example Output

We planned 12 AI agents and shipped 1. It worked better. Sounds crazy, right? But it's a common story.

A client built an AI marketing chatbot. Their initial design had dozens of agents: orchestrator, validators, spam prevention. It failed.

A single agent with tools won. Tasks were tightly coupled. One brain maintained context. Tools were still specialized.

This is the core mistake. People jump to complex multi-agent setups too fast.

Think AI system design as a spectrum:

- Workflows: You control steps.
- Single Agent + Tools: Model decides flow.
- Multi-Agent: Multiple decision-makers.

...

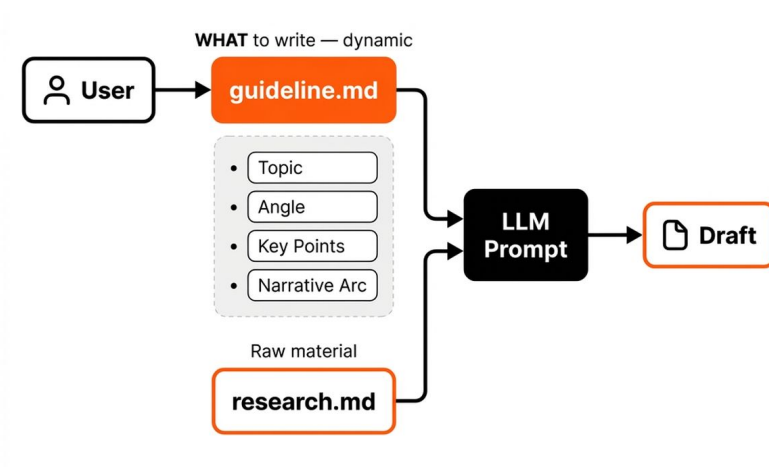
Whole post at: examples/12-agents-to-1-agent

Quick test for AI Slop: <https://eqbench.com/slop-score.html>

Controlling Generation: The Guideline (1)

`guideline.md` = WHAT to write
(dynamic, changes per post)

Defines topic, angle, key points, narrative flow



Example: [examples/from-12-agents-to-1](#)

Controlling Generation: Writing Profiles (2)

Profiles = HOW to write (static, configured once)

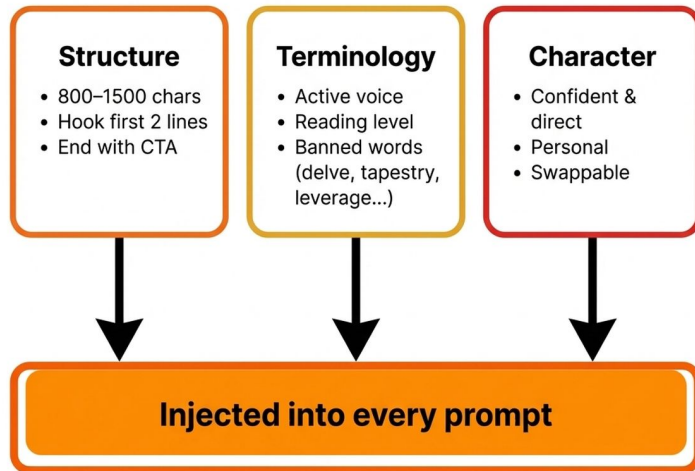
3 Markdown files: Structure, Terminology, Character

Terminology bans 40+ AI slop words and expressions:

- "delve", "tapestry", "vibrant", "landscape", "realm", "embark", "foster"
- ...
- "robust", "scalable", "innovative", "revolutionary", "disruptive"
- "unleash", "beacon", "poised", "unravel", "enrich", "multifaceted"

Full profiles at: [src/writing/profiles](https://github.com/lerocha/llm-recipes/blob/main/src/writing/profiles)

Static — configured once



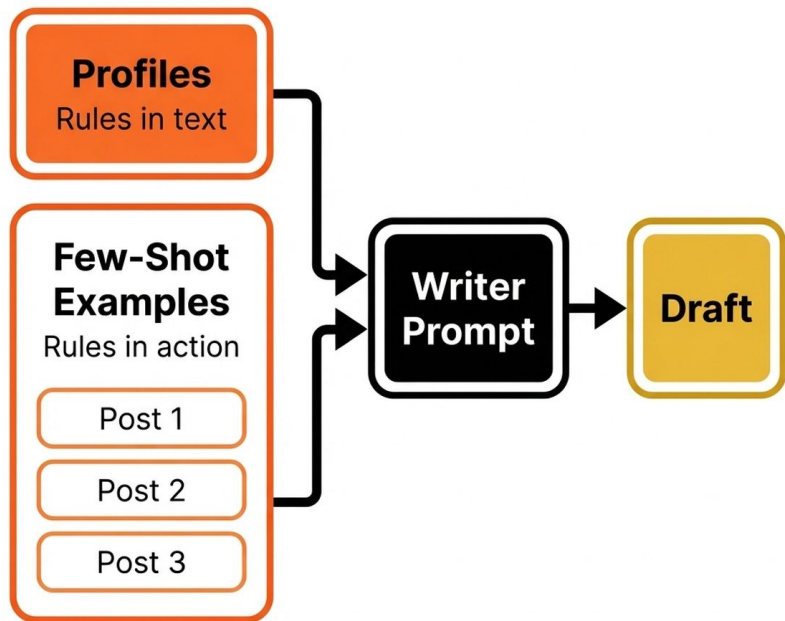
Controlling Generation: Few-Shot Examples (3)

~3 real LinkedIn posts injected into the writer prompt

High-quality and representative

As varied as possible (topic, length, structure, etc.)

Dataset at: [datasets/linkedin_paul_iusztin/index.yaml](https://huggingface.co/datasets/linkedin_paul_iusztin/index.yaml)



Improving Quality: The Evaluator-Optimizer Pattern

Two LLM calls. Different context windows.

The reviewer checks adherence to the:

- Guideline
- Research
- Profiles

The editor (the writer) applies the reviews.

Running in a loop: fixed iterations (~3-4)

- *Keeping each version*
- *Why not using a score?*



Separate LLM calls, distinct prompts, inspectable reviews

Structuring Feedback: How the Review Works

Input: current post

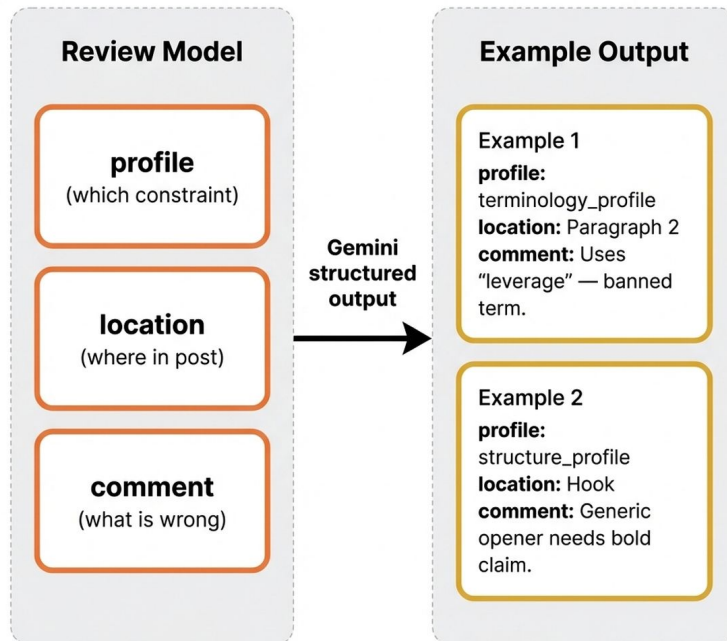
Context: guideline + research + profiles (to check against)

Outputs: Review Pydantic Objects

Reviewer at:

src/writing/app/post_reviewer_handler.py

AI Structured Output Review



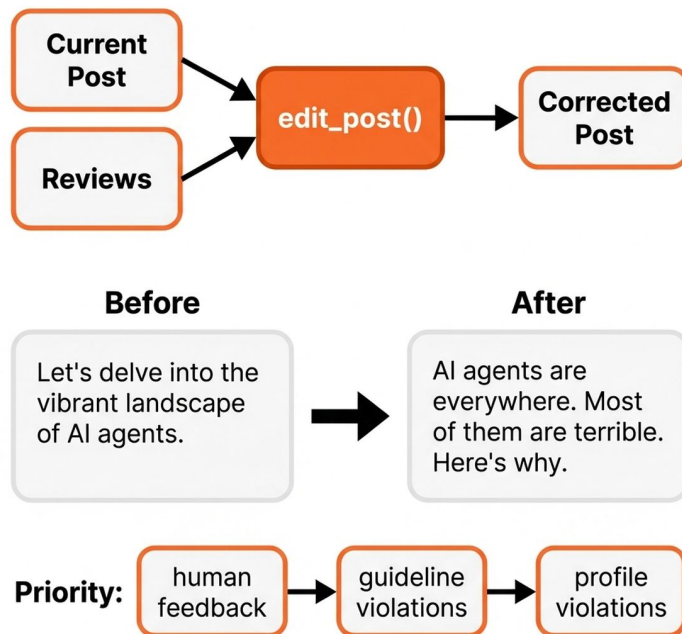
Applying Corrections: How the Edit Works

Input: current post + structured reviews

Context: guideline + research + profiles + few-shot examples (*same as the initial writer*)

Priority: guideline > research > profile violations

Editor at: src/writing/app/post_writer_handler.py



Seeing the Transformation: Before/After

post_v0.md

We planned 12 AI agents. We shipped 1.

Sounds crazy, right? But it's a common story.

A client wanted an AI chatbot for marketing content: emails, SMS, promos. Their initial design had dozens of specialized agents: orchestrator, analyzers, validators, spam prevention.

...

We built an article generator. We started with one agent for research and writing. It broke. Research is exploratory. Writing is constrained. We needed two agents with a clear handoff. Each had its own lean, focused context.

...

Whole post at: examples/from-12-agents-to-1

post_v4.md

We planned 12 AI agents and shipped 1. It worked better. Sounds crazy, right? But it's a common story.

A client built an AI marketing chatbot. Their initial design had dozens of agents: orchestrator, validators, spam prevention. It failed.

A single agent with tools won. Tasks were tightly coupled. One brain maintained context. Tools were still specialized.

...

We built an article generator. We started with one agent for research and writing. It broke.

Research is exploratory. Writing is constrained.

We needed two agents with a clear handoff. Each had its own focused context.

...

DEMO: Running the Writing Workflow Live

3 Levels:

- CLI Command:
`make test-writing-workflow`
- MCP Server Prompt:
`/linkedin-writer:linkedin_post_workflow`
- Skill:
`/write-post`



Recap: What We Built in Part 3

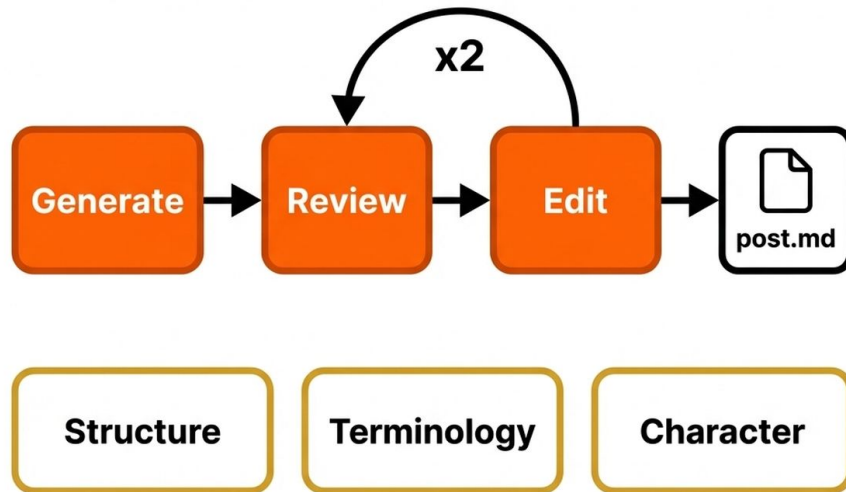
Writing Workflow:

Generate -> Review -> Edit

Controlling Generation:

- Guideline
- Profiles
- Few-shot examples

Run the workflow as an **MCP Server + Skills**



Part 4: Observability: Monitoring & Evaluation

See everything, then measure what matters



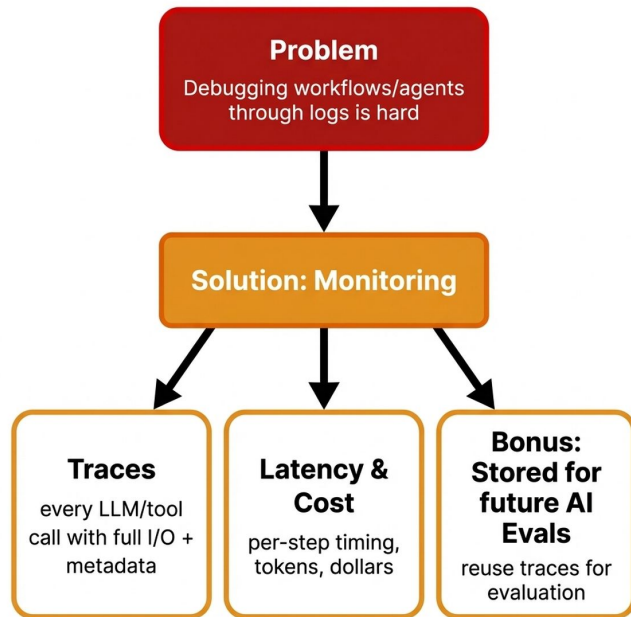
Why You Need Monitoring Since Day 1

Problem: Debugging workflows/agents purely through logs is hard!

Solution: Monitoring

- Traces (every LLM/Tool Call with full I/O + metadata)
- Latency and cost tracking

Bonus: Stores the traces for future AI Evals



DEMO: Exploring the Opik Dashboard

Research Agent

 Thread

 1 Apr 2026, 7:34 PM

 # 12 messages

 2m 4.9s



 +

Messages

Feedback scores

What are the architectural challenges and best practices for integrating LLMs v

Integrating Large Language Models (LLMs) with multiple external APIs in complex, multi-step workflows presents significant architectural challenges. This document explores the most robust best practices to ensure reliability, efficiency, security, and scalability. This integration is crucial for enabling LLMs to perform complex tasks, allowing them to interact with databases, CRM systems, payment gateways, and other specialized services.

Architectural Challenges

- Orchestration and Workflow Management:**
 - Dynamic API Selection:** Determining which API to call, in what order, and with what parameters, based on the user's intent or task, is complex and often requires sophisticated reasoning beyond simple rule-based systems.
 - Multi-step Dependencies:** Workflows often involve a sequence of API calls where the output of one call becomes the input for the next. Managing these dependencies and ensuring proper data flow is challenging, especially when paths diverge or converge based on conditions.
 - Session Management:** Maintaining state across multiple turns of interaction and API calls is difficult, particularly when LLM context windows have limitations.
 - Human-in-the-Loop (HITL):** Integrating points for human intervention or approval within automated workflows adds complexity, requiring mechanisms for persistence, and re-entry into the workflow.
- State Management and Context Window Limitations:**
 - LLMs have a finite context window, meaning they can only "remember" a limited amount of past conversation or information. Managing relevant past information must exceed this limit.
 - External state must be managed outside the LLM, requiring careful design of databases or other state stores to keep track of context from API calls, and user preferences.

Writing Workflow

Thread

1 Apr 2026, 7:21 PM # 20 messages 19m 3.3s \$0.65

Messages

Feedback scores

```
{
  "status": "success",
  "review_iterations": 4,
  "output_path": "/Users/pauliusztin/Documents/01-Projects/Agentic-AI-Engineering-Course/designing-real-world-ai-agents-workshop/example_outputs/ive-switched-from-claude-to-opencode-and-in-loving-it/post.md",
  "message": "Generated LinkedIn post with 4 review/edit iterations. Final post saved to post.md"
}
```

```
{
  "working_dir": "example_outputs/ive-switched-from-claude-to-opencode-and-in-loving-it/post.md",
  "delete_iterations": false
}
```

```
{
  "status": "success",
  "review_iterations": 4,
  "output_path": "/Users/pauliusztin/Documents/01-Projects/Agentic-AI-Engineering-Course/designing-real-world-ai-agents-workshop/example_outputs/ive-switched-from-claude-to-opencode-and-in-loving-it/post.md",
  "message": "Generated LinkedIn post with 4 review/edit iterations. Final post saved to post.md"
}
```

View trace

View tool calls

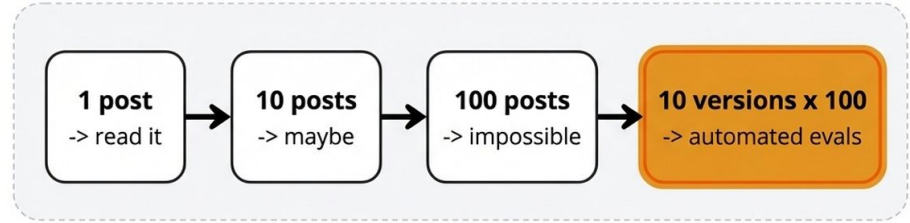
Go to Opik at: comet.com/site/products/opik → Projects

Moving Beyond Vibes: Why Evals Matter

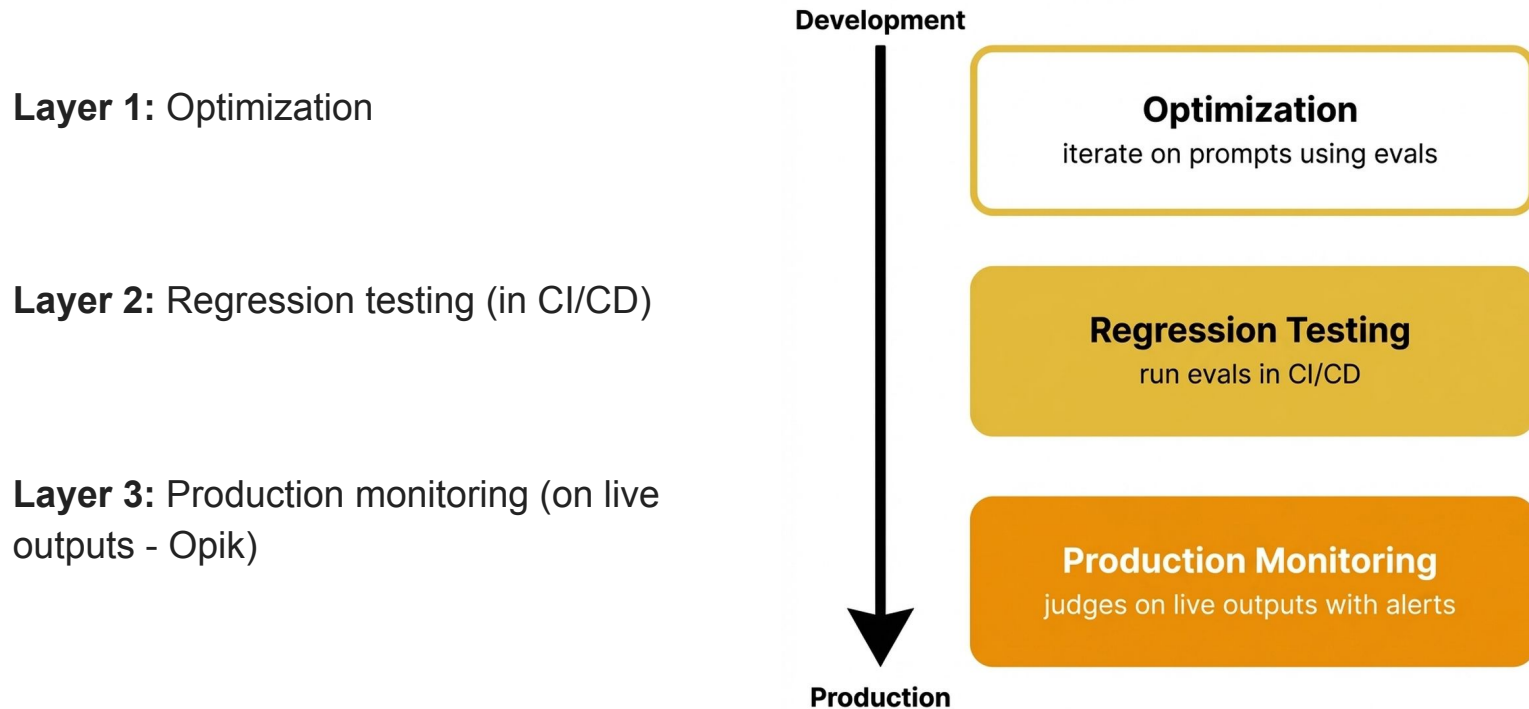
1 post = read it

10 posts = maybe

100 posts = impossible to review and
measure manually



Mapping Where Evals Fit: The Three-Layer Architecture



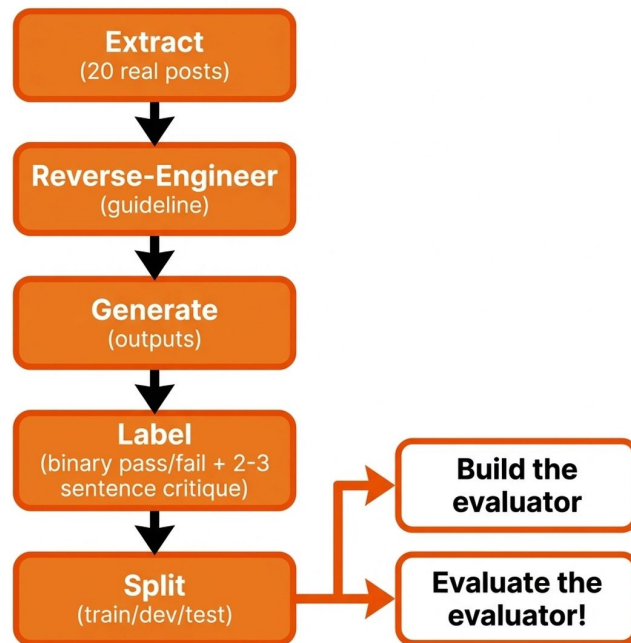
Building an Eval Dataset from Scratch

1. **Extract** 20 real posts
2. **Reverse-engineer** inputs: guideline + research
3. **Generate** outputs
4. **Label**: binary pass/fail + 2-3 sentence critique
5. **Split**: train/dev/test

To:

6. Build the evaluator
7. Evaluate the evaluator!

Dataset at: [datasets/linkedin_paul_iusztin/index.yaml](https://huggingface.co/datasets/linkedin_paul_iusztin/index.yaml)



Automating Judgment: The LLM-as-Judge Pattern

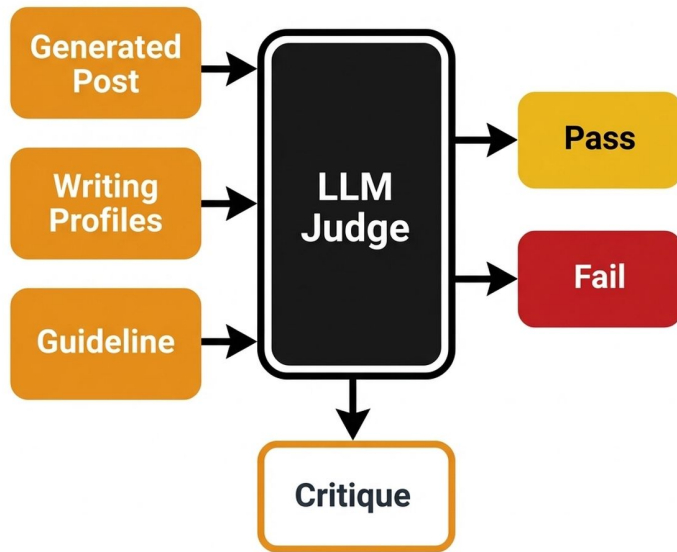
Input: generated post + *profiles* + *research* + *guideline*

Output: returns pass/fail + critique

Few-shot examples (*from the dataset!*):

- Writer Input: Guideline + Research
- Writer Output: LinkedIn Post
- Binary Label: pass/fail
- Critique: 2-3 sentences

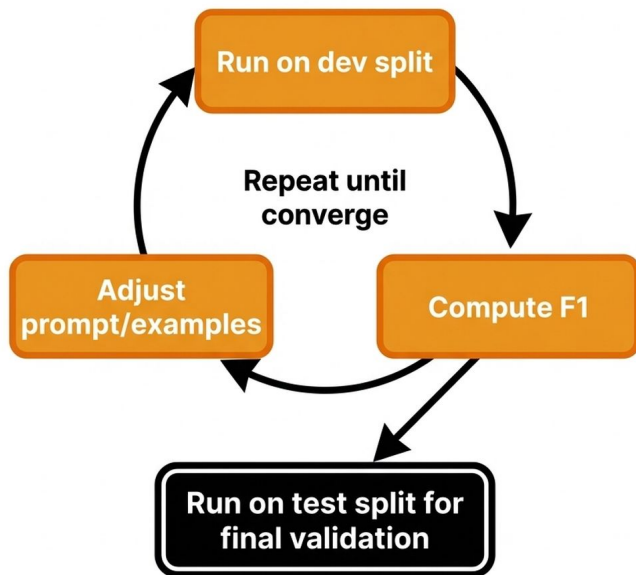
Dataset at: [src/writing/evals/metric.py](https://github.com/justmarkham/llm-recipes/blob/main/src/writing/evals/metric.py)



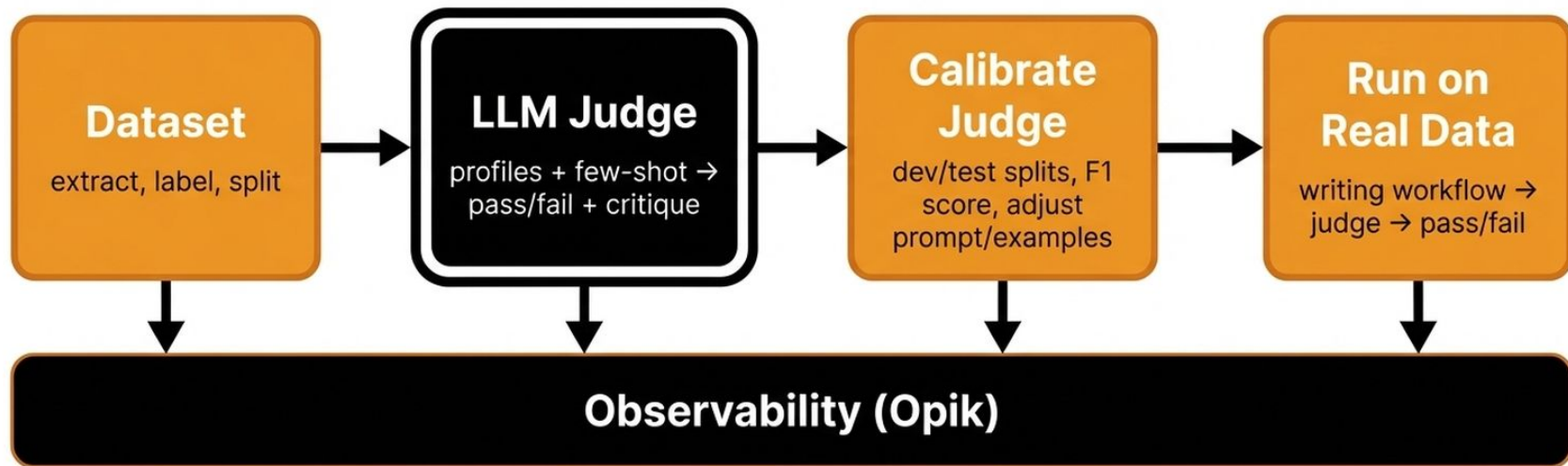
Measuring Judge Reliability

- **LLM Judge:** Binary classifier
- **Goal:** Align LLM Judge with the expert
- **How?**
 - LLM Judge vs. dev/test dataset splits
 - Measure: $F1 = \text{precision} \times \text{recall}$
- **Process:**
 - Run LLM Judge on dev split
 - Compute F1 score
 - Adjust LLM Judge prompt/examples
 - Repeat until converge
 - Run LLM Judge on test split for final validation
- **Ready to run on new data!**

LLM Judge = Binary classifier
Goal: Align LLM Judge with the expert
Metric: $F1 = \text{Precision} \times \text{Recall}$



Wiring It All Together: The Complete Eval Architecture



DEMO: Calibrating the LLM Judge

Running the LLM Judge on the **dev set**:

- Run `make eval-dev`
- F1 score result: Judge vs. expert agreement

Final check on the **test set**:

- Run `make eval-test`
- Compare test F1 to dev F1
- Close = generalizes → Drops = overfit

Go to Opik at: comet.com/site/products/opik → Experiments

DEMO: Production Eval: Simulating Live Traffic

What we test: Posts generated on the fly.

- No expert labels, no F1!
- What we have in real life
- Judge returns pass/fail + critique only

How to run it:

- Run: ``make eval-online``

Go to Opik at: comet.com/site/products/opik → Experiments

DEMO: Full End-to-End Pipeline

SKILL */research-and-write*

Watch for: all traces in one Opik dashboard



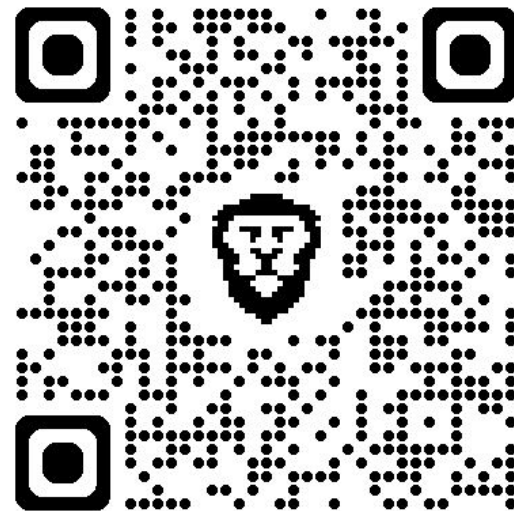
writing-workflow			
Logs Insights Online evaluation Annotation queues			
Threads Traces Spans Search threads... 			
☐	ID -	First message -	Last message -
☐	32a75d51-4139-43fa-b127-37d...	{ "working_dir": "example_outputs/ive-switched-from-claude-code-to-opencode-and-in-L..."	{ "status": "success", "image_path": "/Users/pauliusztin/Documents/01-Projects/Agent..."
☐	49aa7523-9328-4a53-98bd-5a60...	{ "working_dir": "/Users/pauliusztin/Documents/01-Projects/Agentic-AI-Engineering-Co..."	{ "status": "success", "review_iterations": 4, "output_path": "/Users/pauliusztin/Do..."
☐	512c1953-99a8-4ed2-8810-c0be0...	{ "working_dir": "/Users/pauliusztin/Documents/01-Projects/Agentic-AI-Engineering-Co..."	{ "status": "success", "review_iterations": 2, "output_path": "/Users/pauliusztin/Do..."

Go to Opik at: comet.com/site/products/opik → Projects

Open up the GitHub repository

Run everything yourself

Read the code



<https://github.com/iusztinpaul/designing-real-world-ai-agents-workshop/>

Whenever you're ready, go deeper with the:

Agentic AI Engineering Course

Goal? Design and build production-ready AI agents

34 lessons. Three end-to-end portfolio projects. A certificate. And a Discord community.

Rated 5/5 by 300+ students.

The first 6 lessons are free.

<https://academy.towardsai.net/courses/agent-engineering>

