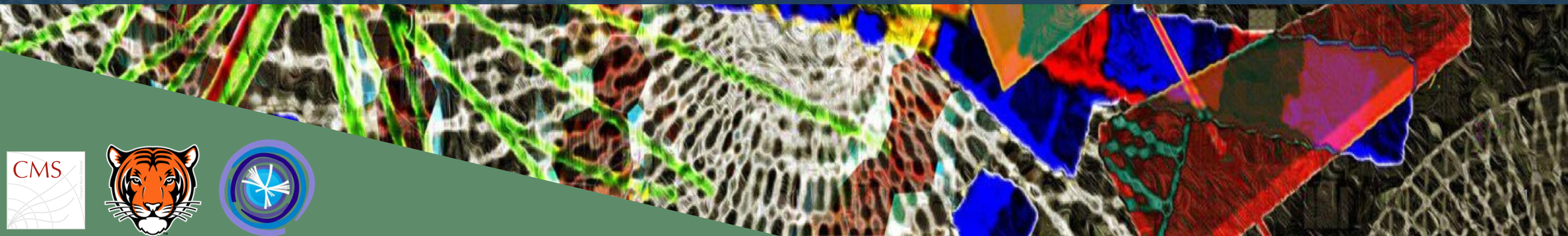
An AI Assistant for CMS Combine

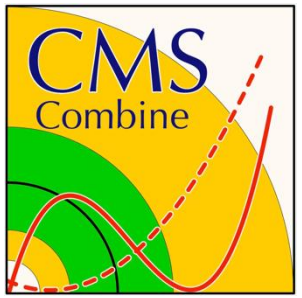
Massimiliano Galli

PyHEP.dev

7 September 2026



Motivation



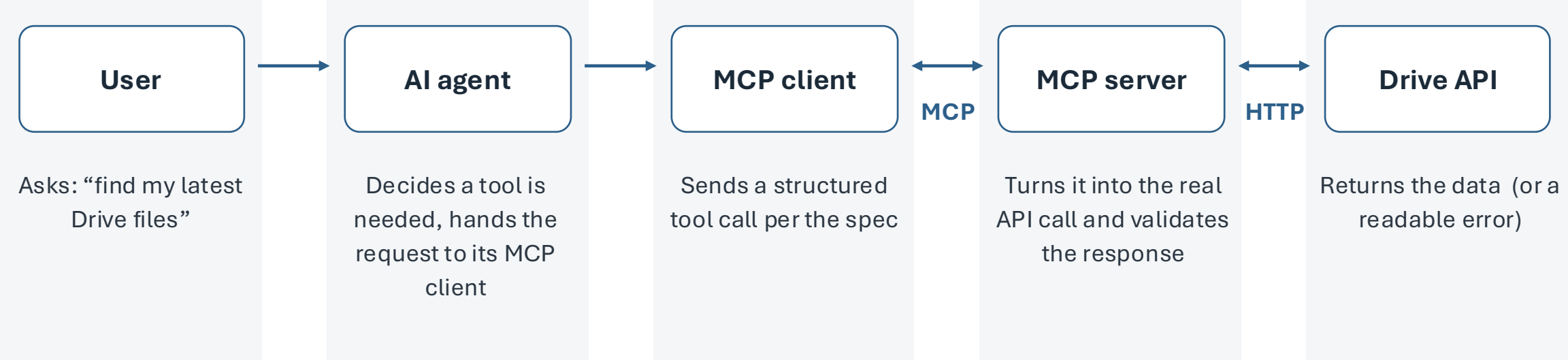
- **CMS Combine** is the statistical inference tool used by ~90% of the CMS analyses, but:
  - **Steep learning curve**
  - Difficult API with a **CLI** and **several command line arguments**
- Build an **AI assistant** that:
  - **Answer Combine questions** with citations (grounded in documentation, paper, code, forum)
  - **Run Combine commands** to reproduce, diagnose, confirm results

How?

- Follow the same idea (and, ideally, infrastructure) used by a colleague in IT to build the lumi-assistant for CERN services
- The idea: build a pluggable wrapper around LLM providers, with a **skill system** that indicates the agent how to behave depending on the situation and **MCP servers** that provide access to the Combine-related documentation
- **Advantages**:
  - The modular approach makes it easier to use different clients depending on one's subscribtions
  - Agent can directly rely on Combine's CLI if available
  - Minimal effort

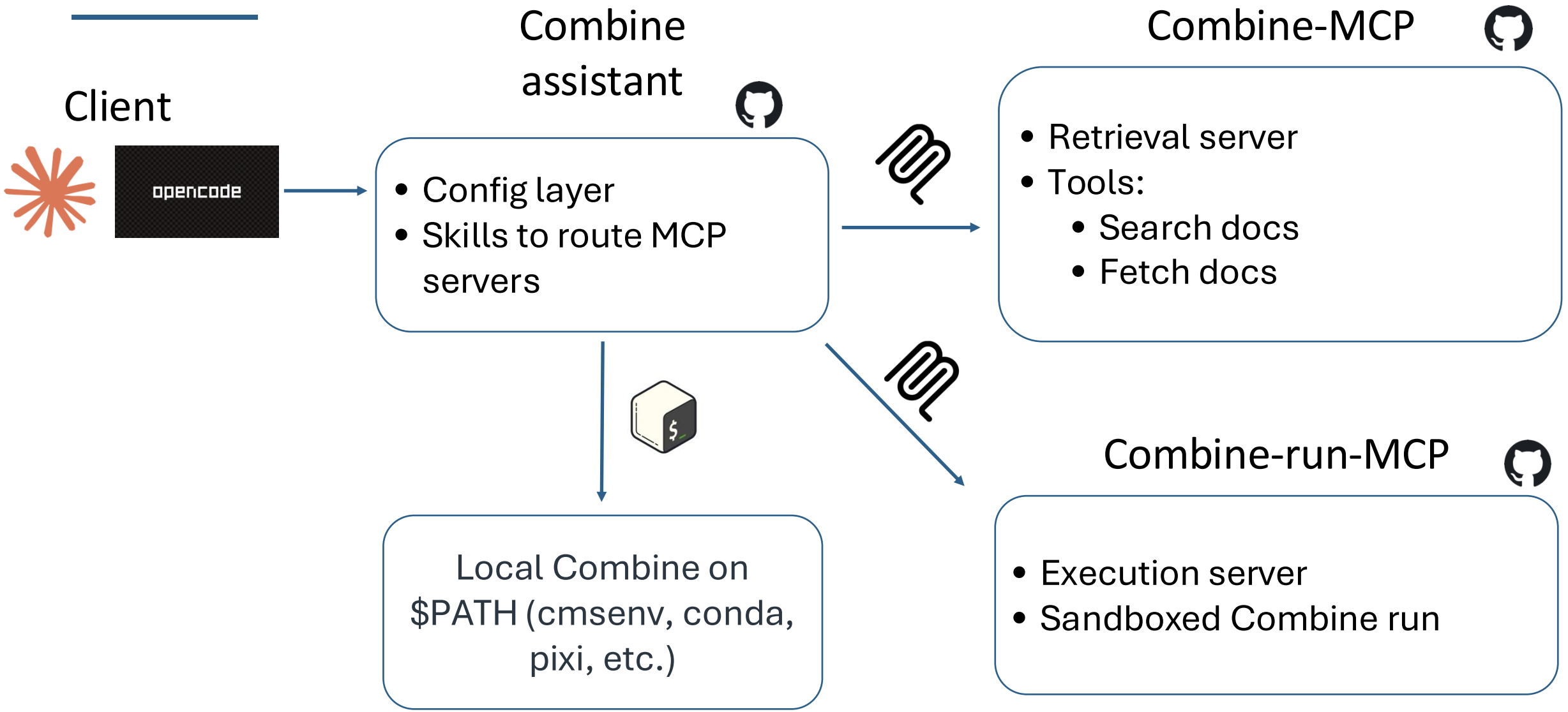
What is MCP?

- **MCP (Model Context Protocol)**: an open standard for connecting an AI agent to tools and data
- Each server publishes a **list of tools** with a **schema** for their arguments (argument names, types and which are required) so the agent knows exactly how to call a tool
- Add a server, gain a capability



Example adapted from quickchat.ai/post/mcp-explained

Architecture



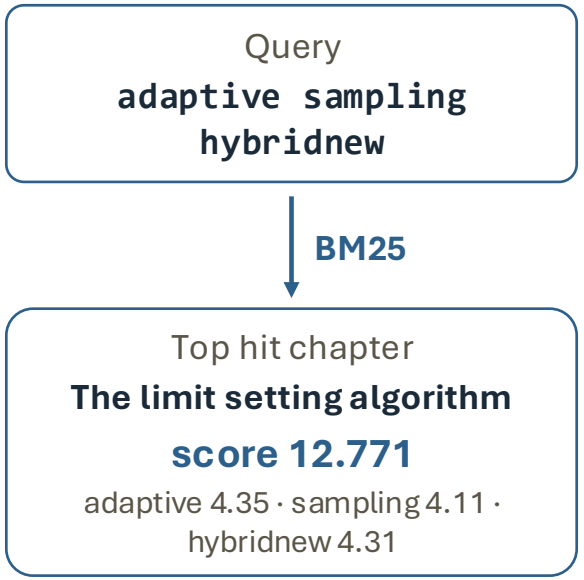
Architecture

Combine-MCP

- Retrieval server
- Tools:
  - Search docs
  - Fetch docs

Combine-MCP (retrieval server)

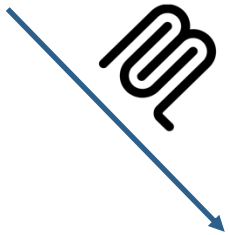
- Search/fetch documentation over five sources (tools: **search_docs**, **fetch_doc**)
- Ranked with BM25 algorithm
  - Scores a document against a query by summing, over each query term, how often that term appears in the document, weighted by how rare that term is across the whole corpus
  - Only lexical, no semantics
- Deployed with PaaS at combine-mcp-git-combine-mcp.app.cern.ch/mcp



Source	Content
combine-docs	Official MkDocs documentation: CLI flags, how-to, tutorials
combine-paper	arXiv:2404.06614v2: methodology and definitions
combine-code	Pinned source tree: the implementation
combine-forum	cms-talk Statistics category (2022→): errors, real-world edge cases
combine-hypernews	Archived HyperNews (pre-2022), lower priority

Architecture

Local Combine on $PATH (cmsenv, conda, pixi, etc.)



Combine-run-MCP

- Execution server
- Sandboxed Combine run

Execution

- In certain situations (when debugging a command that fails) we want the agent to directly **run Combine** commands
- Use skills to setup priority:
  - First, try to **run with user's installation** if combine executable is available on PATH
  - If not available, run the command remotely via **combine-run-mcp** server
- Deployed on PaaS at https://combine-run-mcp-git-combine-mcp.app.cern.ch/mcp
- Note: users should ideally always go through the first option when datacards and ROOT files are large

Architecture

Client

opencode

Combine assistant

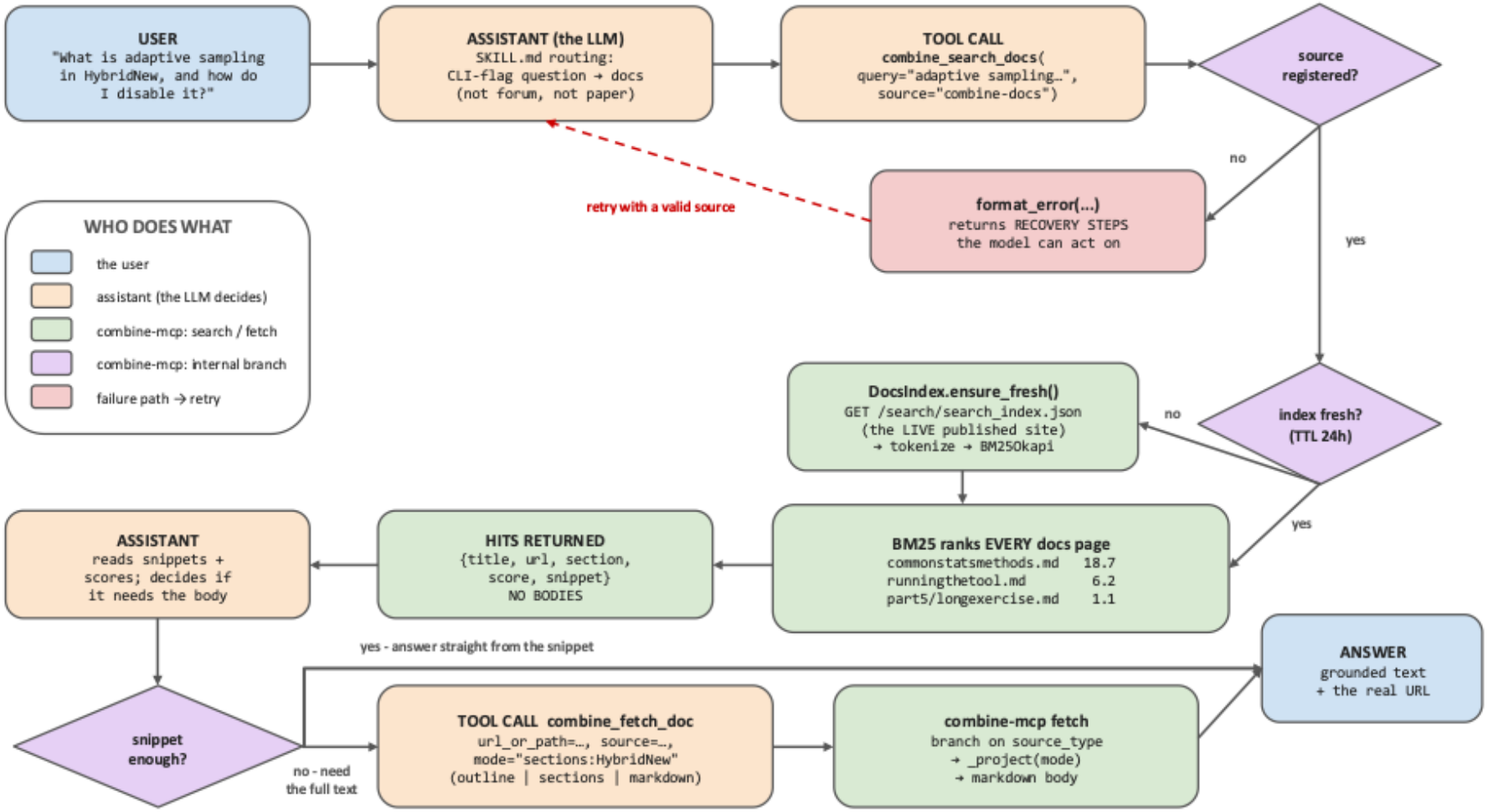
- Config layer
- Skills to route MCP servers

Combine-assistant (assembly layer)

- **Config layer** that turns the different pieces into a **usable agent** and distributes it
- Two clients (for now): **opencode** and **Claude code**
- **SKILL.md** is the domain brain:
  - routes each question to the right source
  - reads BM25 scores intelligently
  - enforces anti-hallucination
  - drives the reproduce → diagnose → fix loop for execution

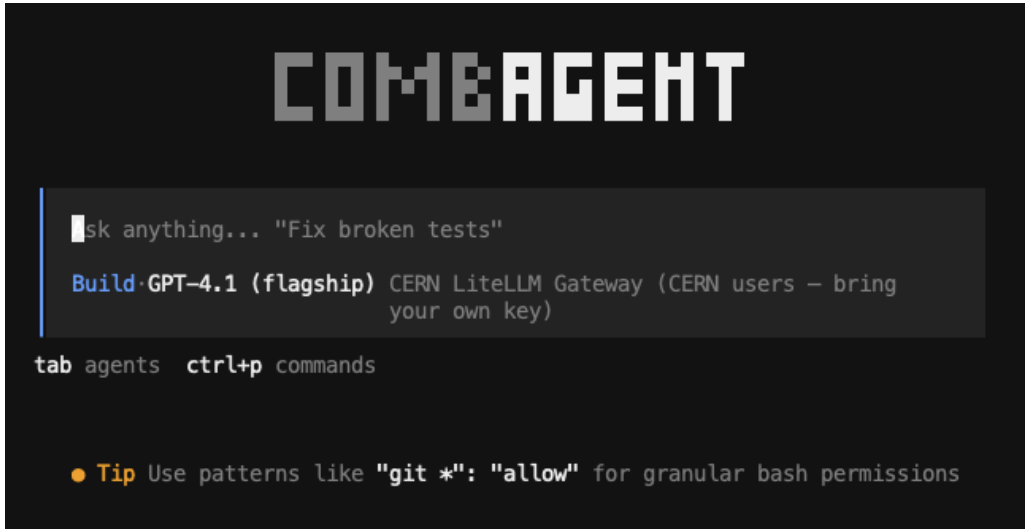
opencode

Example (assistant + Combine-MCP)



Distribution and deployment

- **Modularity** makes the agent usable in different ways
- With a **Claude subscription** (and a local installation of Combine), one can simply clone the combine-assistant and point Claude to it
- However, since most of Combine users work on **Lxplus**, we also **ship on CVMFS CAT area** (together with a fork of opencode, called **combagent** – instructions)



Work-in-progress with CERN facilities

- Ideally would like to have **CERN facilities** as main gateways for powerful LLMs
- Tested and implemented on the CERN litellm gateway https://llmgw-litellm.web.cern.ch/
  - Provided endpoints and tokens for many powerful models, including GPT-5.5
  - However, **will be decommisioned soon** (end of September) as not sustainable for many users
- **New AI gateway** https://aigw.cern.ch/ is still work in progress and DOES NOT provide the same models as before
  - Not possible at the moment to use the hosted models due to not suitable serving
  - Work in progress and coordinating work with them, using this agent as benchmark and test for the platform they are developing

Testing and evaluating

- Testsuite of **8 questions** (private to cms-analysis gitlab) working on others to extend the tests
- Metrics:
  - **Answer quality**:
    - Factual correctness: compare answer to “golden answer” (provided by us) using an LLM judge, without penalizing extra details in the answer (recall mode)
    - Rubric judge (same LLM as before) that rates separately correctness, completeness and grounding; more similar to human judge
  - **Trajectory**:
    - Deterministic (no judge involved)
    - Check the order of used tools, if the expected source was searched, expected docs retrieved, etc.

“What is adaptive sampling in Combine's HybridNew method, and how do I control or disable it from the command line?”

Agent answer

I’ll ground this in the Combine docs/code so the option names and behavior are exact.

Adaptive sampling in `HybridNew` is Combine’s automatic toy-statistics refinement during a limit scan: at each tested signal-strength point, it can generate additional toys until the estimated `CLs` uncertainty reaches the requested absolute accuracy, or until the limit precision cannot improve further. This is why `HybridNew` output can show multiple `CLs = ... +/- ...` lines for the same `r` value. The limit scan itself is also adaptive in `r`, stopping based on `rAbsAcc` or `rRelAcc`. See the Combine docs on computing limits with toys and accuracy.

Main controls:

```
combine datacard.txt -M HybridNew --LHCmode LHC-limits --clsAcc 0.005
```

- `--clsAcc X`: target absolute accuracy for the `CLs` value during the limit search. The docs state the default is `0.5%`; tighter precision costs a lot because improving the accuracy by a factor `N` needs about `N^2` more toys.
- `-T N` or `--toysH N`: minimum number of toys generated for each point. Default is `500`.
- `--rAbsAcc X` and `--rRelAcc X`: stopping criteria for the limit value itself, defaulting to `0.1` and `0.05`, respectively.
- `--iterations I`: for complex models, internally split the toys over several `HybridNew` instances; total toys are `I * N` where `N` is from `-T`.

To disable adaptive sampling, set:

```
combine datacard.txt -M HybridNew --LHCmode LHC-limits --clsAcc 0
```

This is the documented recipe for fixed-toy grid production, for example:

```
combine datacard.txt -M HybridNew --LHCmode LHC-limits \
  --singlePoint X --saveToys --saveHybridResult -T 500 --clsAcc 0
```

In that mode, Combine uses the fixed number of toys you requested with `-T` instead of adaptively adding toys to satisfy a `CLs` precision target. This is recommended in the docs when producing grids for complex models that you will later merge and read back with `--readHybridResults --grid=merged.root`; see Complex models.

Golden answer

By default, HybridNew uses adaptive sampling: toys are thrown in batches and the procedure stops as soon as the statistical uncertainty on CLs (or p_μ) reaches the requested accuracy, or once the current estimate is clearly far from the CL contour of interest. This avoids wasting computation on points whose answer is already statistically unambiguous.

Two command-line options govern the behaviour:

- `--toysH N` (or `-T N`): minimum number of toys thrown per batch (default 500). Increase this for higher CL levels (e.g. 99%).
- `--clsAcc X`: target absolute accuracy on CLs (default 0.005).

To disable adaptive sampling and throw a fixed number of toys, set `--clsAcc 0`. The algorithm then runs exactly `--iterations` × `T` toys without early termination. This is the standard pattern when building a grid of test-statistic distributions for complex models:

```
combine datacard.txt -M HybridNew --LHCmode LHC-limits \
    --singlePoint X --saveToys --saveHybridResult \
    -T 500 --clsAcc 0
```

Factual correctness	Rubric correctness	Rubric completeness	Rubric grounding	Rubric overall
0.65	5.0	5.0	5.0	5.0

Agent running gpt-5.5

Judge LLM gpt-4.1

“What is adaptive sampling in Combine's HybridNew method, and how do I control or disable it from the command line?”

Trajectory:

1. skill: **combine**
2. `search_docs` **combine-docs** — "HybridNew adaptive sampling --noAdaptiveSampling --clsAcc --singlePoint"
   → *tutorial_stat_routines/stat_routines/, part3/commonstatsmethods/, part3/commonstatsmethods/, +2 more*
3. `search_docs` **combine-code** — "adaptive sampling HybridNew"
   → *interface/HybridNew.h, interface/MarkovChainMC.h, interface/SimplerLikelihoodRatioTestStat.h, +2 more*
4. `fetch_doc` **combine-code** — `interface/HybridNew.h` (markdown)
5. `fetch_doc` **combine-docs** — `part3/commonstatsmethods/` (sections: Complex models)
6. `fetch_doc` **combine-docs** — `tutorial_stat_routines/stat_routines/` (sections: The limit setting algorithm)
7. `search_docs` **combine-docs** — "--clsAcc --clsAccuracy HybridNew adaptive"
   → *part3/commonstatsmethods/, part3/commonstatsmethods/, tutorial_stat_routines/stat_routines/, +5 more*
8. `search_docs` **combine-code** — "clsAcc noAdaptive HybridNew"
   → *interface/HybridNew.h, interface/SimplerLikelihoodRatioTestStat.h, interface/Combine.h, +4 more*
9. `fetch_doc` **combine-docs** — `part3/commonstatsmethods/` (sections: Accuracy)
10. `fetch_doc` **combine-code** — `.../blob/v10.6.0/interface/HybridNew.h` on GitHub (markdown)
11. `fetch_doc` **combine-code** — `interface/HybridNew.h` (outline)
12. `fetch_doc` **combine-docs** — `part3/commonstatsmethods/` (sections: Computing limits with toys)

Next steps and conclusions

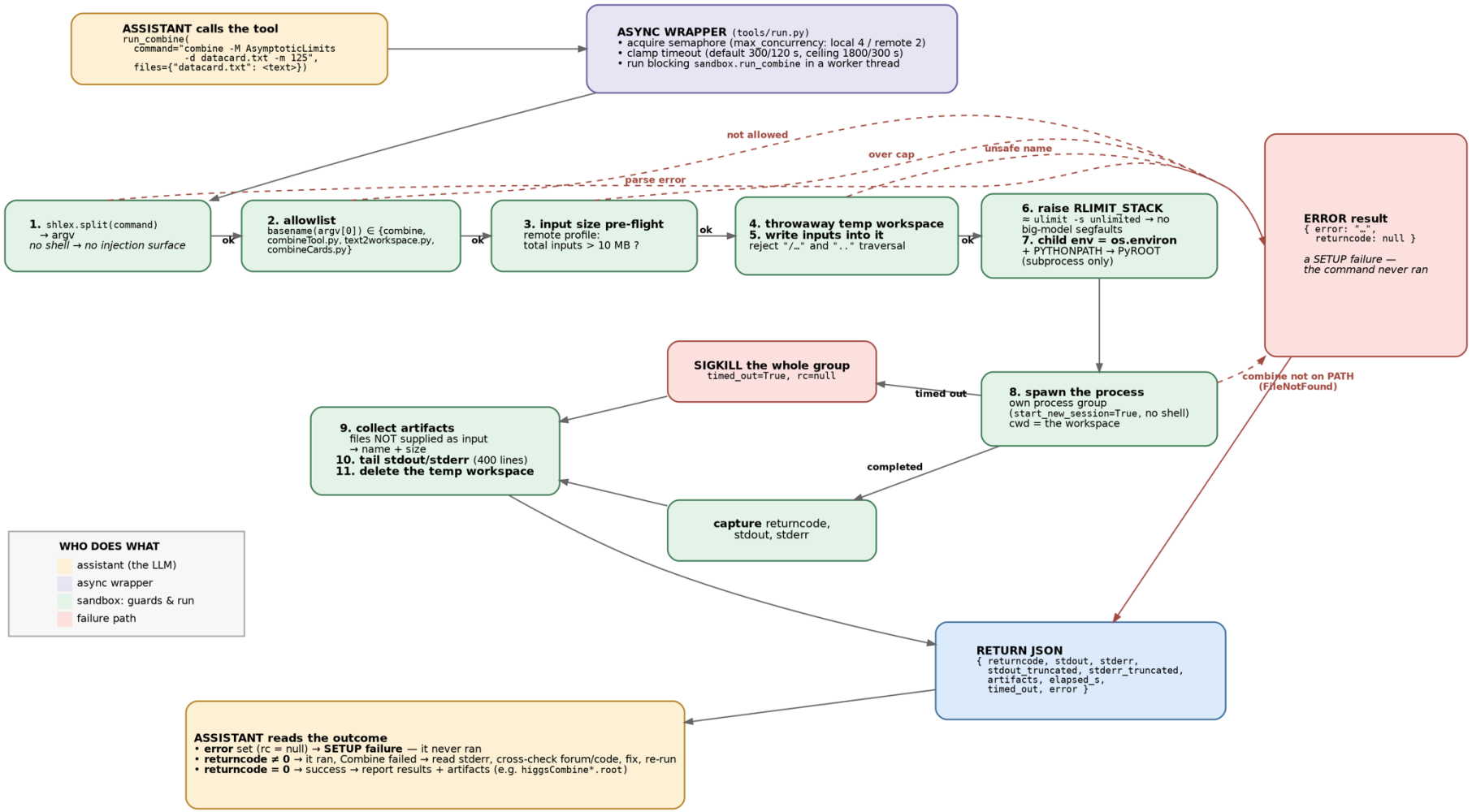
- Setup an AI agent for CMS Combine using **skills+MCP** servers
- **Next steps**:
  - Extend testing to spot weaknesses in both docs retrieval and answer generation (**more ways to test?**)
  - Additional MCPs:
    - ROOT-mcp, if useful (to add knowledge and tools to manipulate ROOT files)
    - Possibly a memory-MCP, ideally shared across users to increase precision of the agent
  - Setup the agent to run also in CI during the datacard (statistical model format) reviews
  - Follow-up with CERN IT to use infrastructure and self-hosted models as much as possible

Backup

How?

- First attempt: RAG-based Q/A bot built on GPT 4.1 (GitHub, website accessible only from CERN network or VPN) – however:
  - More difficult to implement
- Then: conversations with Giovanni Guerrieri from IT (+ the fact that I could have just re-used most of the code developed by them for CERN's lumi-assistant) convinced me to try a more modular and agent-like approach based on MCP servers and skills

Combine-run-MCP (execution server)



Models/Credentials

Provider	Models	Notes
litellm (will be decommissioned soon)	gpt-5.5 (default), gpt-5.6-sol/terra/luna, gpt-5.4, gpt-5.3-chat-latest, gpt-5.2, gpt-5, gpt-4.1, gpt-4.1-nano, gpt-oss-20b, mistral-small-latest	CERN LiteLLM gateway, LITELLM_API_KEY (shared key on EOS via lumi-api-access). All GPT models: tool calls ✅, vision ✅, 128k–1M context. Being decommissioned; has twice exhausted its upstream OpenAI credits (HTTP 429), currently back up. Also has the prompt-cache bug — an identical repeated prompt can hang or return empty, hence the cache-bust nonce in the evals.
aigw (testing, WIP)	gpt-oss-20b, hf-qwen3-32b-awq	CERN AI Gateway, per-user AIGW_API_KEY (self-service UI → pick team → "All Team Models"). The designated successor, but neither model can drive the agent: hf-qwen3-32b-awq returns no tool_calls under tool_choice="auto" and is served with a 16k context (< our ~29k system prompt); gpt-oss-20b aborts multi-turn runs with harmony parse errors. No vision on either. Both reported upstream with reproducers. Lives on aigw-migration, not in the shipped v0.4.7.
nrp	kimi, qwen3, qwen3-small (+ gpt-oss on the branch)	NRP Nautilus, free for researchers, NRP_API_KEY. All verified working end-to-end. kimi = 262k context, tool calls in streaming and non-streaming, and real vision (correctly read the NLL-scan plot) — currently the eval agent. qwen3 needs enable_thinking:false for clean JSON (handled automatically by score.py) — currently the eval judge. gpt-oss works here while the same model fails on aigw, which is the evidence those are serving-config bugs. Non-CERN service.
anthropic	claude-sonnet-5, claude-opus-4-8	Public Anthropic API, your own ANTHROPIC_API_KEY. Tool calls ✅, vision ✅, 200k context. The reliable paid fallback; also usable through Claude Code as an alternative client (--engine claude).
cern-vm	qwen2.5-coder:7b	Self-hosted Ollama on Combine-bot.cern.ch, no key at all. CPU-only, so an agentic question takes minutes. Zero-credential fallback, not a daily driver. CERN network only.

Backup: how BM25 ranks a page

- Per query term, BM25 multiplies how often the term appears in the page (TF) by how rare it is across the corpus (IDF) — so distinctive words like `clsAcc` dominate and ubiquitous ones like `the` barely count
- Two corrections keep it honest: term frequency saturates, so the 50th occurrence adds far less than the 2nd, and scores are normalised by page length, so a long page can't win just by being long

$$\text{contribution} = \text{IDF} \cdot \frac{f(k_1+1)}{f + k_1(1 - b + b \cdot |D|/\text{avgdl})} \quad \text{with } k_1 = 1.5,\ b = 0.75$$

Live docs corpus: 386 documents, average length 344 tokens · query "adaptive sampling HybridNew" · winning page: The limit setting algorithm (405 tokens)

token	in how many docs	IDF
adaptive	3 / 386	4.697
sampling	4 / 386	4.443
hybridnew	30 / 386	2.459

token	tf	contribution
adaptive	1	4.347
sampling	1	4.112
hybridnew	4	4.312
		12.771 ← the page's score

- hybridnew appears four times yet contributes no more than the single-occurrence words: saturation flattens the extra hits, and its low IDF caps what they were worth — so a page cannot climb by repeating the obvious keyword

Reproducible against the live docs with bm25_demo.py