How to build and maintain an agentic evaluation: eight steps from claim to reported score

24 November 2026 · 10 min read · blogs.iictp.org/agentic-evaluation-how-to-build-maintain-explained

Short answer: an agentic evaluation gives an AI agent a task in a controlled environment, lets it act, and grades the final state of that environment, so the score depends on the task, the agent harness, the verifier and the traces, not only the model. A Hugging Face guide published on 23 November 2026 walks through eight steps from defining the claim to comparing models, and shows how a grader that checked only one number could be fooled by a shortcut script.

TL;DR The short version

- An agentic evaluation gives an agent a task in a controlled environment and grades the final state, so the score depends on the task, harness, verifier and traces, not just the model.
- A Hugging Face guide sets out eight steps from a narrow claim to comparing models, and shows how a weak grader can be fooled by a shortcut script.
- Treat an evaluation as software with tests: the reference should pass, a null agent should fail and a cheat should fail.

Read online, listen to it, or change the text size: https://blogs.iictp.org/agentic-evaluation-how-to-build-maintain-explained

What is an agentic evaluation?

The guide, by Avijit Ghosh, Ben Burtenshaw and Irene Solaiman, describes three eras: question-and-answer benchmarks such as MMLU and GPQA, agentic benchmarks such as SWE-bench and WebArena that check what an agent changed in an environment, and team or swarm evaluations that score what a group of agents produced. It focuses on the second. Because the agent uses tools, every part of the surrounding software can change the score: the post notes an audit of Terminal-Bench in which the model was fixed and only the agent harness varied.

The anatomy of an evaluation

- **Task:** the instruction given to the agent.
- **Agent harness:** the wrapper around the model and its tools, running inside a pinned environment.
- **Rollout:** one full attempt, possibly many commands, from start to finish. The evaluation harness runs several rollouts per task.
- **Verifier:** grades the environment's final state.
- **Trace and score:** each rollout leaves a trace; traces and scores together are the published evidence.

How it works: the eight steps

1. Define the claim, kept deliberately narrow.
2. Find an existing task to reuse.
3. Run it unchanged to get a baseline.
4. Adapt it to your domain.
5. Check the evaluation itself.
6. Read the traces.
7. Report the score.
8. Compare models.

Worked example and results

- The example is a physics-lab agent fitting peaks in a Raman spectrum of graphene and computing a defect (D-to-G) ratio. It reuses the open Harbor task format and Terminal-Bench's raman-fitting task, runs the Terminus-2 harness with gpt-oss-120b, and uses Hugging Face Jobs, Sandboxes and Inference Providers.
- On the original task gpt-oss-120b passed 0 of 3 rollouts. In two it declared completion before seeing relevant output, and in one it could not read a data file that uses decimal commas. The score alone could not distinguish these from a wrong fit; the traces could.
- The authors built eight synthetic spectra with known parameters, then tested the grader with the reference solution, a do-nothing agent and a shortcut script. The shortcut read peak heights and guessed widths, yet returned the expected ratio on all eight tasks, so a ratio-only check would have given full marks. They added checks on each fitted peak.

Why release decisions matter

- The guide says the Hub hosts about 3,500 evaluation leaderboards and that eval datasets grew about 16-fold in a year to about 17,000.
- It cites that 98.2% of model-benchmark scores are reported by only one party, that Artificial Analysis says 60% of its Intelligence Index v5 (announced October 2026) rests on private test sets or held-out answers, and that Google DeepMind piloted double-blind evaluations. It argues privacy alone does not guarantee quality.

Limitations and open questions

- The worked example is deliberately small, eight development tasks, and the guide says broader coverage should follow real variation.
- Statistics in the guide come from the authors and cited third parties, not from our reproduction.
- Team and swarm evaluations are explicitly out of scope.

What it means for engineers and AI teams

Treat the evaluation as software that needs tests: check that the reference passes, a null agent fails and a cheat fails. Read traces before trusting scores. See how to evaluate an LLM application, GitHub Copilot evals and testing vendor claims with your own evals.

Frequently asked questions

- **What is an agentic evaluation?** A test where an agent works in a pinned environment and a verifier grades the final state, with traces kept as evidence.
- **Why can the harness change the score?** Because the agent's tools and loop are part of the system being measured.
- **Why read traces?** Scores cannot tell early stopping or a data-reading failure from a wrong answer.
- **Should the grader be tested?** Yes. Run a reference solution, a no-op and a shortcut to confirm it passes and fails correctly.

Related reading

- How to evaluate an LLM application
- GitHub Copilot evals in public preview
- Gemini 4 Carbon: vendor claims and your own evals
- Local LLM Practitioner

Sources and currency

Sources were read on 24 November 2026. Products, rules and standards change, so check the linked originals before you rely on any detail.

> Related credential: Local LLM Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. Avijit Ghosh, Ben Burtenshaw and Irene Solaiman, Hugging Face. How to build and maintain an agentic evaluation. 23 November 2026. huggingface.co/blog/agentic-evals.

Published by IICTP, an independent technical credential body, on 24 November 2026. The current version of this post is at https://blogs.iictp.org/agentic-evaluation-how-to-build-maintain-explained. More guides: blogs.iictp.org.