

BENCH_15_09

Comparative benchmark between two AI assistants on real tasks
OpenFreedom (OF) · OpenClaw (OC) — F1/F4 phases on configurable models

OF V.12.19 · OpenClaw 2026.9.4

15 September 2026

How the test was run

Seven tests with prompts written the way a real user would write them: no indication of where to save, no hint about the problems to find. The tests cover artifact generation (SaaS project, webapp, launch plan), scenario reasoning, repair with a constrained destination and — new in this session — a test measured on LLM calls. Order: OF first, OC then, one task at a time (the two runners share the test machine).

For each test we saved: the exact prompt, the full answer, consumption (billed tokens), the files produced and the substantial verification where applicable (compilation, JSON validity, counts).

Vanilla platforms: both in their official version, with no modification for the test. In this session the context growth was intentional: the cache was not cleared and the sessions were not renewed between tests — it is the condition that made the flow's behaviour observable under accumulated memory.

Same LLM, the same for both: deepseek-flash (id deepseek-v4-flash). In OF the F1 and F4 phases run on that model; in OC it is its gateway's model. Neither platform had a dedicated or more powerful model.

Sub-agents deliberately blocked in OF: for this suite OF was kept on a single agent, in a single session — no delegation to sub-agents. The comparison therefore measures the single flow in one session, not an orchestration: it is the condition that makes the calls figure readable.

Machine and installations

- **Test machine:** all-in-one PC, Linux. KittProxy hosts the OF instance on which the OF tasks run; pop-os hosts the OC runner and the owner's OF instance. One task at a time.
- **OF V.12.19** — official webui, F1→gate→F4 flow with stellar memory; phases on a configurable model.
- **OC 2026.9.4** — official gateway on KittProxy; tasks tagged [uid:1] + via kittproxy (needed so its guardrail authorises the action).
- **Price list** (billed tokens, \$/million, deepseek-flash off-peak): new input 0.15 · cache 0.003 · output 0.60.

The test is repeatable — and a method limit, declared

- Suite and prompts are versioned: bench-2026-09-14-4settori/suite-*.json and bench-UFFICIALE-2026-09-15/prompt/<ID>.txt (39 prompts, exact text + check).
- Re-run: run_of_bench.py --suite <suite> --solo <ID> --out results-of-<tag>.json (OF, on KittProxy) and run_oc_bench.py --suite <suite> --solo <ID> --tag <tag> --out results-oc-<tag>.json (OC, from pop-os).
- Results: results-*.json and log/*.log; per-run detail in run-dettaglio-2026-09-15.md.
- **Declared limit:** unlike the 20/08 bench, the KittProxy session was not reset between tests. On the final task (CRM, not in the table) this produced a rewrite of the previous task: the theme results of the KittProxy tests must be read with this caveat. The consumption figures remain valid as a measure of the work done, not of maximum capacity.

Full table

test	OF	OC (with the marker)	OC/OF gap
T1 • SAAS1 SaaS booking platform — project from scratch	OF 119.1 s · 3 calls · \$0.01553 · 48,338 B	OC 167.2 s · 15 calls · \$0.02838 · 60,673 B	+40% time · +83% cost
T2 • BUS1 Digital company analysis — scenario reasoning	OF 18.9 s · 2 calls · \$0.00381 · 7,192 B	OC 559.2 s · 44 calls · \$0.08835 · 46,481 B	+2,858% time · +2,219% cost
T3 • PZ1 Urban square redevelopment — technical project	OF 152.4 s · 2 calls · \$0.01393 · 9,371 B	OC 147.9 s · 5 calls · \$0.01662 · 40,346 B	−3% time · +19% cost
T4 • APP1 Student mobile app launch — operational plan	OF 114.2 s · 2 calls · \$0.01399 · 34,998 B (pop-os 88 s)	OC 309.7 s · 23 calls · \$0.04202 · ~924 lines	+171% time · +200% cost
T5 • WEB1 Agricultural market webapp — navigable artifact	OF 42.8 s · 2 calls · \$0.00958 · 33,595 B	OC 439.0 s · 22 calls · \$0.07745 · 108,564 B + 35,432 B	+926% time · +709% cost
T6 • FIXCREA Delivery to the requested folder — destination	OF 33.6 s · 3 calls · \$0.00597 · folder + file inside + source intact + gate OK	OC 156.4 s · 13 calls · \$0.02568 · folder used + file	+366% time · +330% cost
T7 • TRIAGE1 IT ticket triage — four working artifacts	OF 136.8 s · 2 calls · \$0.01288 · 4/4 compliant	OC 792.6 s · 39 calls · \$0.12586 · 4/4 compliant	+479% time · +877% cost · +1,850% calls

The individual tests

T1 • SAAS1 — SaaS booking platform — project from scratch

Design from scratch the architecture of a SaaS platform for managing bookings for small professional practices. The system must support user registration, authentication, roles and permissions, calendar, bookings, email notifications, payments, conflict handling and an admin panel. Define application

architecture, data model, main APIs, user flows, authentication strategy, error handling, security, caching and scalability. Compare at least two possible architectures and choose the most suitable one, justifying the choice. Conclude with an implementation roadmap ordered by priority and dependencies. Save the report in a text file.

What is being tested: designing the complete architecture of a booking platform: data model, flows, constraints. No hint, no indication of where to save.

OC/OF gap: +40% time · +83% cost

OF 119.1 s · 3 calls · \$0.01553 · 48,338 B | OC 167.2 s · 15 calls · \$0.02838 · 60,673 B

OC started from an already-present v1.1 (reuse, not generation)

Delivered: OF a design report (48,338 B); OC a larger document (60,673 B) but based on an existing v1.1.

Opinion: both deliver a coherent design artifact; the comparison on size is not informative, because OC did not start from scratch.

T2 · BUS1 — Digital company analysis — scenario reasoning

Analyse the following scenario: a small digital company has 3 products, 12 employees, fixed monthly costs of 38,000 euro, an average gross margin of 72%, 420,000 euro of annual revenue and forecast growth of 18%. Product A generates 55% of revenue but grows only 4%; B generates 30% and grows 35%; C generates 15% and grows 60% but has a 41% margin. The company can invest at most 120,000 euro over the next 12 months. Analyse sustainability, risks, revenue concentration and opportunities. Build at least three capital-allocation scenarios, estimate the economic impact of each...

What is being tested: scenario reasoning on incomplete business data, with constrained allocation.

OC/OF gap: +2,858% time · +2,219% cost

OF 18.9 s · 2 calls · \$0.00381 · 7,192 B | OC 559.2 s · 44 calls · \$0.08835 · 46,481 B

OC revised an existing v1.0; the delta is dominated by this case

Delivered: OF a lean analysis (7,192 B); OC an extended revision (46,481 B).

Opinion: the gap (+2,858%) measures OC's revision, not its capacity: 44 calls and 9 minutes to amend an existing document. OF, in 2 calls and 19 seconds, produced the requested analysis.

T3 · PZ1 — Urban square redevelopment — technical project

Imagine redeveloping an 8,000 m² urban square in an Italian city. The project must increase greenery and shading, keep accessibility for people with disabilities, reduce car traffic without eliminating it completely, create spaces for commerce and socialising, manage rainwater drainage and respect a maximum budget of 2.5 million euro. Propose a complete concept, split the area by function, define design priorities, materials and infrastructure, estimate costs and maintenance qualitatively, analyse conflicts between objectives and propose a phased implementation sequence. Conclude by stating which trade-offs you consider unavoidable. Save the result in a text file.

What is being tested: design with declared trade-offs. The only test where OC was faster.

OC/OF gap: -3% time · +19% cost

OF 152.4 s · 2 calls · \$0.01393 · 9,371 B | OC 147.9 s · 5 calls · \$0.01662 · 40,346 B

Shifted tie: faster, more expensive

Delivered: OF riqualificazione_piazza_8000mq.txt (9,371 B, opened and verified: concept with vision, guiding principles and declared budget); OC 40,346 B.

Opinion: correct, readable delivery from both; OC's 4 extra calls are more structure, not more mandatory content.

T4 · APP1 — Student mobile app launch — operational plan

You must launch from scratch a mobile app that helps university students organise study, exams and deadlines. You have an initial budget of 50,000 euro and a team of 2 developers, 1 designer and 1 marketing person. Define the minimum viable product, target and positioning, monetisation model, 6-month technical roadmap, user-acquisition strategy, metrics to monitor, launch plan and economic scenarios. Also identify the main product and market risks and establish which experiments to run before spending a significant part of the budget. Give a final recommendation on what to build, what to postpone and what not to do. Deliver the result in a text file.

What is being tested: launch plan with a final recommendation on what to build and what not to.

OC/OF gap: +171% time · +200% cost

OF 114.2 s · 2 calls · \$0.01399 · 34,998 B (pop-os 88 s) | OC 309.7 s · 23 calls · \$0.04202 · ~924 lines

First test with substantial verification of the content

Delivered: OF piano_lancio_app_studio_universitari.txt (34,998 B, verified: budget, team and 6-month horizon at the top); OC ~924 lines.

Opinion: both complete; OF with 2 calls, OC with 23 — same operational outcome.

T5 · WEB1 — Agricultural market webapp — navigable artifact

Create a complete webapp for managing a local farmers' market. It must let users browse the producers present, see the available products, filter by category and buy products for pickup on a given date. Producers must have a panel to update availability and quantities. Administrators must be able to manage producers, orders, pickup time slots and statistics. Design a modern, responsive and accessible interface. Implement search, filters, cart, simulated checkout, authentication, role management, notifications and a dashboard. Use realistic demo data and make the app immediately navigable without external configuration.

What is being tested: a complete, self-sufficient application, with no external configuration. It tests the production of a real artifact.

OC/OF gap: +926% time · +709% cost

OF 42.8 s · 2 calls · \$0.00958 · 33,595 B | OC 439.0 s · 22 calls · \$0.07745 · 108,564 B + 35,432 B

OC from scratch: no pre-existing file, the purest comparison

Delivered: OF a single self-sufficient HTML file (33,595 B, verified: doctype and viewport, openable with no configuration); OC 108,564 B plus a test harness (35,432 B).
Opinion: OF delivers the requested artifact at the minimum sufficient level; OC adds automated tests the prompt did not ask for — real value, but paid with 22 calls against 2.

T6 · FIXCREA — Delivery to the requested folder — destination

analyse and repair this file benchmark_buggy_app.py which you find in the attachments. Create the folder /home/umberto/Scrivania/file riparato_benchmark_buggy_app.py and save the result inside. The source file must only be read and not touched.

What is being tested: repairing a file and saving it into a folder with the exact name, without touching the source. It tests the DESTINATION, not the speed.

OC/OF gap: +366% time · +330% cost

OF 33.6 s · 3 calls · \$0.00597 · folder + file inside + source intact + gate OK | OC 156.4 s · 13 calls · \$0.02568 · folder used + file

The test that closed the most serious defect of the day

Delivered: OF the folder file riparato_benchmark_buggy_app.py created, the repaired file inside, the source intact (md5 unchanged) and the gate OK; OC the same folder used.

Opinion: the test that closed the most serious defect of the day: destination respected and source untouched, in 3 calls against 13.

T7 · TRIAGE1 — IT ticket triage — four working artifacts

You are an AI Operations Engineer at an ICT company that handles technical support for business customers. Design and implement a small automatic IT ticket triage system. The system must: receive tickets in JSON format; automatically classify category, priority and severity; distinguish incident, service request and security incident; identify tickets that require human escalation; propose an operational procedure (runbook) consistent with the problem; avoid automatic actions when there is insufficient information; produce a structured, validatable JSON output. Create these artifacts: 1. ticket_triage.py — a working implementation; 2. knowledge_base.md — a knowledge base with at least 12 cases; 3. runbook.md — operational procedures for the main cases; 4. test_cases.json — at least 25 test tickets, including ambiguous, contradictory and deliberately misleading cases. The system must be deterministic where possible, easily extensible and accompanied by automated tests. Do not limit yourself to describing the architecture: produce the working artifacts.

What is being tested: building a triage system with four verifiable artifacts: code that compiles, valid JSON, knowledge base and runbook. The test of the CALLS.

OC/OF gap: +479% time · +877% cost · +1,850% calls

OF 136.8 s · 2 calls · \$0.01288 · 4/4 compliant | OC 792.6 s · 39 calls · \$0.12586 · 4/4 compliant

Same compliance, 2 calls against 39

Delivered: OF 4 artifacts out of 4 verified — ticket_triage.py compiles (19 functions), test_cases.json valid with 28 tickets, knowledge base 38 cases, runbook 18 sections. OC: 4 out of 4 plus tests/, schema and README (33 tickets, 26 cases, ~172 KB).

Opinion: identical compliance and beyond the minimums; OF wins on calls (2 against 39) and

on the knowledge-base content (38 cases in a quarter of the weight), OC wins on literal adherence (the required automated tests, which OF omitted).

Conclusions

- OF always delivers in the five comparison tests (5 out of 5) and is the cheapest and fastest column: 447.4 s and \$0.05684 in total against 1,623 s and \$0.25282 for OC.
- The gap is largest where OC rebuilds from scratch (WEB1: +926% time) and smallest where OC starts from existing artifacts: in the two SAAS1 and BUS1 runs, 41 of its 44/22 calls were reuse or revision — not a pure comparison, and that must be said.
- The test on calls (T7/TRIAGE1) weighs the most: same compliance (4 artifacts out of 4, beyond the minimums), 2 calls against 39. OC's knowledge base weighs 4 times OF's and contains 32% fewer cases: the big file does not buy content.