

# REPORT FINALE · SUITE BENCH

## 7 PROVE

# BENCH\_15\_09

Benchmark comparativo tra due assistenti AI su compiti reali  
OpenFreedom (OF) · OpenClaw (OC) — fasi F1/F4 su modelli configurabili

OF V.12.19 · OpenClaw 2026.9.4

15 settembre 2026

## Come si è svolta la prova

Sette prove con prompt scritti come li scriverebbe un utente reale: nessuna indicazione di dove salvare, nessun suggerimento sui problemi da trovare. Le prove coprono generazione di artefatti (progetto SaaS, webapp, piano di lancio), ragionamento su scenario, riparazione con destinazione vincolata e — novità di questa sessione — una prova misurata sulle **chiamate LLM**. Ordine: **OF prima, OC poi**, un task alla volta (i due runner condividono la macchina di test).

Per ogni prova sono stati salvati: prompt esatto, risposta completa, consumi (token fatturati), file prodotti e la verifica sostanziale dove applicabile (compilazione, validità JSON, conteggi).

**Piattaforme vanilla:** entrambe in versione ufficiale, senza alcuna modifica per la prova. In questa sessione la **crescita di contesto è stata voluta**: la cache non è stata cancellata e le sessioni non sono state rinnovate fra le prove — è la condizione che ha reso osservabile il comportamento del flusso sotto memoria accumulata.

**Stessa LLM, la stessa per entrambe: deepseek-flash** (id deepseek-v4-flash). In OF le fasi F1 e F4 girano su quel modello; in OC è il modello del suo gateway. Nessuna piattaforma ha avuto un modello dedicato o più potente.

**Sub-agenti bloccati di proposito in OF:** per questa suite OF è stato tenuto su **un solo agente, in un'unica sessione** — nessuna delega a sub-agenti. Il confronto misura quindi il **flusso singolo** in una sessione, non un'orchestrazione: è la condizione che rende leggibile il dato sulle chiamate.

## Macchina e installazioni

- **Macchina di test:** PC all-in-one, Linux. **KittProxy** ospita l'istanza OF su cui girano i task OF; **pop-os** ospita il runner OC e l'istanza OF del proprietario. Un task alla volta.
- **OF V.12.19** — webui ufficiale, flusso F1→gate→F4 con memoria stellare; fasi su modello configurabile.
- **OC 2026.9.4** — gateway ufficiale su KittProxy; i task marcati con [uid:1] + via kittproxy (necessari perché il suo guardrail autorizzi l'azione).
- **Listino** (token fatturati, \$/milione, deepseek-flash off-peak): input nuovo 0,15 · cache 0,003 · output 0,60.

La prova è ripetibile — e un limite di metodo, dichiarato

- Suite e prompt sono versionati: bench-2026-09-14-4settori/suite-\*.json e bench-UFFICIALE-2026-09-15/prompt/<ID>.txt (39 prompt, testo esatto + check).
- Rilancio: run\_of\_bench.py --suite <suite> --solo <ID> --out results-of-<tag>.json (OF, su KittProxy) e run\_oc\_bench.py --suite <suite> --solo <ID> --tag <tag> --out results-oc-<tag>.json (OC, da pop-os).
- Risultati: results-\*.json e log/\*.log; dettaglio per run in run-dettaglio-2026-09-15.md.
- **Limite dichiarato:** a differenza del bench del 20/08, la **sessione di KittProxy non è stata azzerata fra le prove**. Sul task finale (CRM, non in tabella) questo ha prodotto la riscrittura del task precedente: i risultati di *tema* delle prove su KittProxy vanno letti con questa riserva. I consumi restano validi come misura del lavoro svolto, non della capacità massima.

Tabella completa

| prova                                                                    | OF                                                                                 | OC (col marcatore)                                     | divario OC/OF                                |
|--------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------|----------------------------------------------|
| <b>T1 · SAAS1</b><br>Piattaforma SaaS di prenotazioni — progetto da zero | OF 119,1 s · 3 ch · \$0,01553 · 48.338 B                                           | OC 167,2 s · 15 ch · \$0,02838 · 60.673 B              | +40% tempo · +83% costo                      |
| <b>T2 · BUS1</b><br>Analisi azienda digitale — ragionamento su scenario  | OF 18,9 s · 2 ch · \$0,00381 · 7.192 B                                             | OC 559,2 s · 44 ch · \$0,08835 · 46.481 B              | +2.858% tempo · +2.219% costo                |
| <b>T3 · PZ1</b><br>Riqualificazione piazza urbana — progetto tecnico     | OF 152,4 s · 2 ch · \$0,01393 · 9.371 B                                            | OC 147,9 s · 5 ch · \$0,01662 · 40.346 B               | −3% tempo · +19% costo                       |
| <b>T4 · APP1</b><br>Lancio app mobile studenti — piano operativo         | OF 114,2 s · 2 ch · \$0,01399 · 34.998 B (pop-os 88 s)                             | OC 309,7 s · 23 ch · \$0,04202 · ~924 righe            | +171% tempo · +200% costo                    |
| <b>T5 · WEB1</b><br>Webapp mercato agricolo — artefatto navigabile       | OF 42,8 s · 2 ch · \$0,00958 · 33.595 B                                            | OC 439,0 s · 22 ch · \$0,07745 · 108.564 B + 35.432 B  | +926% tempo · +709% costo                    |
| <b>T6 · FIXCREA</b><br>Consegna nella cartella richiesta — destinazione  | OF 33,6 s · 3 ch · \$0,00597 · cartella + file dentro + sorgente intatto + gate OK | OC 156,4 s · 13 ch · \$0,02568 · cartella usata + file | +366% tempo · +330% costo                    |
| <b>T7 · TRIAGE1</b><br>Triage ticket IT — quattro artefatti funzionanti  | OF 136,8 s · 2 ch · \$0,01288 · 4/4 conformi                                       | OC 792,6 s · 39 ch · \$0,12586 · 4/4 conformi          | +479% tempo · +877% costo · +1.850% chiamate |

I singoli test

T1 · SAAS1 — Piattaforma SaaS di prenotazioni — progetto da zero

Progetta da zero l'architettura di una piattaforma SaaS per la gestione di prenotazioni per piccoli studi professionali. Il sistema deve supportare registrazione utenti, autenticazione, ruoli e permessi, calendario, prenotazioni, notifiche email, pagamenti, gestione dei conflitti e pannello amministrativo. Definisci architettura applicativa, modello dati, API principali, flussi utente, strategia di autenticazione, gestione degli errori, sicurezza, caching e scalabilità. Confronta almeno due possibili

architetture e scegli quella piu adatta motivando la scelta. Concludi con una roadmap di implementazione ordinata per priorit  e dipendenze. Salva il report in un file di testo.

**Cosa si sta testando:** Progettare l'architettura completa di una piattaforma di prenotazione: modello dati, flussi, vincoli. Nessun suggerimento, nessuna indicazione di dove salvare.

Divario OC/OF: **+40% tempo · +83% costo**

| OF                                       | OC (col marcatore)                        |
|------------------------------------------|-------------------------------------------|
| OF 119,1 s · 3 ch · \$0,01553 · 48.338 B | OC 167,2 s · 15 ch · \$0,02838 · 60.673 B |

OC partiva da una v1.1 gia' presente (riuso, non generazione)

**Consegnato:** OF un report di progetto (48.338 B); OC un documento piu' ampio (60.673 B) ma su una v1.1 *gi  esistente*.

**Parere:** entrambi consegnano un artefatto di progetto coerente; il confronto sulla dimensione non e' informativo, perche' OC non partiva da zero.

## T2 · BUS1 — Analisi azienda digitale — ragionamento su scenario

Analizza il seguente scenario: una piccola azienda digitale ha 3 prodotti, 12 dipendenti, costi fissi mensili di 38.000 euro, margine lordo medio del 72%, 420.000 euro di fatturato annuo e una crescita prevista del 18%. Il prodotto A genera il 55% del fatturato ma cresce solo del 4%; B genera il 30% e cresce del 35%; C genera il 15% e cresce del 60% ma ha margine del 41%. L'azienda puo investire al massimo 120.000 euro nei prossimi 12 mesi. Analizza sostenibilit , rischi, concentrazione del fatturato e opportunit . Costruisci almeno tre scenari di allocazione del capitale, stima l'impatto economico di ciascuno e formula una raccomandazione finale motivata. rendi tutto in un report.

**Cosa si sta testando:** Analisi di uno scenario aziendale con raccomandazioni operative. Testa il ragionamento, non la produzione di file.

Divario OC/OF: **+2.858% tempo · +2.219% costo**

| OF                                     | OC (col marcatore)                        |
|----------------------------------------|-------------------------------------------|
| OF 18,9 s · 2 ch · \$0,00381 · 7.192 B | OC 559,2 s · 44 ch · \$0,08835 · 46.481 B |

OC ha rivisto una v1.0 esistente; il delta e' dominato da questo caso

**Consegnato:** OF un'analisi asciutta (7.192 B); OC una revisione estesa (46.481 B).

**Parere:** il divario (+2.858%) misura la *revisione* di OC, non la capacita': 44 chiamate e 9 minuti per emendare un documento esistente. OF, in 2 chiamate e 19 secondi, ha prodotto l'analisi richiesta.

## T3 · PZ1 — Riqualificazione piazza urbana — progetto tecnico

Immagina di dover riqualificare una piazza urbana di 8.000 m2 in una citta italiana. Il progetto deve aumentare verde e ombreggiamento, mantenere l'accessibilit  per persone con disabilit , ridurre il traffico automobilistico senza eliminarlo completamente, creare spazi per commercio e socialit , gestire il drenaggio delle acque piovane e rispettare un budget massimo di 2,5 milioni di euro. Proponi un concept completo, suddividi la superficie per funzioni, definisci priorit  progettuali, materiali e infrastrutture, stima qualitativamente costi e manutenzione, analizza i conflitti tra gli obiettivi e proponi una sequenza di realizzazione in fasi. Concludi indicando quali compromessi ritieni inevitabili. Salva in un file di testo il risultato

**Cosa si sta testando:** Progettazione con compromessi dichiarati. L'unico test dove OC e' stato piu' veloce.

Divario OC/OF: **-3% tempo · +19% costo**

| OF                                      | OC (col marcatore)                       |
|-----------------------------------------|------------------------------------------|
| OF 152,4 s · 2 ch · \$0,01393 · 9.371 B | OC 147,9 s · 5 ch · \$0,01662 · 40.346 B |

Pareggio spostato: piu' veloce, piu' caro

**Consegnato:** OF riqualificazione\_piazza\_8000mq.txt (9.371 B, aperto e verificato: concept con visione, principi guida e budget dichiarato); OC 40.346 B.

**Parere:** consegna corretta e leggibile da entrambi; le 4 volte di OC sono struttura in piu', non contenuto obbligatorio in piu'.

## T4 · APP1 — Lancio app mobile studenti — piano operativo

Devi lanciare da zero un'app mobile che aiuta studenti universitari a organizzare studio, esami e scadenze. Disponi di un budget iniziale di 50.000 euro e di un team composto da 2 sviluppatori, 1 designer e 1 persona marketing. Definisci il prodotto minimo lanciabile, target e posizionamento, modello di monetizzazione, roadmap tecnica di 6 mesi, strategia di acquisizione utenti, metriche da monitorare, piano di lancio e scenari economici. Individua inoltre i principali rischi di prodotto e mercato e stabilisci quali esperimenti eseguire prima di spendere una parte significativa del budget. Dai una raccomandazione finale su cosa costruire, cosa rimandare e cosa non fare. consegna il risultato in un file di testo

**Cosa si sta testando:** Piano di lancio con raccomandazione finale su cosa costruire e cosa no.

Divario OC/OF: **+171% tempo · +200% costo**

| OF                                                     | OC (col marcatore)                          |
|--------------------------------------------------------|---------------------------------------------|
| OF 114,2 s · 2 ch · \$0,01399 · 34.998 B (pop-os 88 s) | OC 309,7 s · 23 ch · \$0,04202 · ~924 righe |

Prima prova con verifica sostanziale sul contenuto

**Consegnato:** OF piano\_lancio\_app\_studio\_universitari.txt (34.998 B, verificato: budget, team e orizzonte 6 mesi in testa); OC ~924 righe.

**Parere:** entrambi completi; OF con 2 chiamate, OC con 23 — stesso esito operativo.

## T5 · WEB1 — Webapp mercato agricolo — artefatto navigabile

Crea una webapp completa per la gestione di un mercato agricolo locale. Deve permettere agli utenti di consultare i produttori presenti, vedere i prodotti disponibili, filtrare per categoria e acquistare prodotti per il ritiro in una determinata data. I produttori devono avere un pannello per aggiornare disponibilita e quantita. Gli amministratori devono poter gestire produttori, ordini, fasce orarie di ritiro e statistiche. Progetta un'interfaccia moderna, responsive e accessibile. Implementa ricerca, filtri, carrello, checkout simulato, autenticazione, gestione dei ruoli, notifiche e dashboard. Usa dati demo realistici e fai in modo che l'app sia immediatamente navigabile senza configurazioni esterne.

**Cosa si sta testando:** Applicazione completa, autosufficiente, senza configurazioni esterne. Testa la produzione di un artefatto vero.

Divario OC/OF: **+926% tempo · +709% costo**

| OF | OC (col marcatore) |
|----|--------------------|
|----|--------------------|

|                                         |                                                       |
|-----------------------------------------|-------------------------------------------------------|
| OF 42,8 s · 2 ch · \$0,00958 · 33.595 B | OC 439,0 s · 22 ch · \$0,07745 · 108.564 B + 35.432 B |
|-----------------------------------------|-------------------------------------------------------|

OC da zero: nessun file preesistente, il confronto piu' puro

**Consegnato:** OF un unico file HTML autosufficiente (33.595 B, verificato: doctype e viewport, apribile senza configurazioni); OC 108.564 B *piu'* un harness di test (35.432 B).

**Parere:** OF consegna l'artefatto richiesto al minimo sufficiente; OC aggiunge test automatici che il prompt non chiedeva — valore reale, ma pagato 22 chiamate contro 2.

## T6 · FIXCREA — Consegna nella cartella richiesta — destinazione

analizza e ripara questo file `benchmark_buggy_app.py` che trovi in allegati. Crea la cartella `/home/umberto/Scrivania/file riparato` `benchmark_buggy_app.py` e salva dentro il risultato. Il file sorgente deve essere solo letto e non toccato

**Cosa si sta testando:** Riparare un file e salvarlo in una cartella dal nome esatto, senza toccare il sorgente. Testa la DESTINAZIONE, non la velocita'.

Divario OC/OF: **+366% tempo · +330% costo**

| OF                                                                                 | OC (col marcatore)                                     |
|------------------------------------------------------------------------------------|--------------------------------------------------------|
| OF 33,6 s · 3 ch · \$0,00597 · cartella + file dentro + sorgente intatto + gate OK | OC 156,4 s · 13 ch · \$0,02568 · cartella usata + file |

Il test che ha chiuso il difetto piu' grave della giornata

**Consegnato:** OF la **cartella** `file_riparato_benchmark_buggy_app.py` creata, il file riparato **dentro**, il **sorgente intatto** (md5 invariato) e il gate OK; OC la stessa cartella usata.

**Parere:** la prova che ha chiuso il difetto piu' grave della giornata: destinazione rispettata e partenza non toccata, in 3 chiamate contro 13.

## T7 · TRIAGE1 — Triage ticket IT — quattro artefatti funzionanti

Sei un AI Operations Engineer di un'azienda ICT che gestisce l'assistenza tecnica per clienti business. Progetta e implementa un piccolo sistema di triage automatico dei ticket IT. Il sistema deve: ricevere ticket in formato JSON; classificare automaticamente categoria, priorita e severita; distinguere incident, service request e security incident; individuare ticket che richiedono escalation umana; proporre una procedura operativa (runbook) coerente con il problema; evitare azioni automatiche quando mancano informazioni sufficienti; produrre un output JSON strutturato e validabile. Crea questi artefatti: 1. `ticket_triage.py` - implementazione funzionante; 2. `knowledge_base.md` - knowledge base con almeno 12 casistiche; 3. `runbook.md` - procedure operative per le casistiche principali; 4. `test_cases.json` - almeno 25 ticket di test, inclusi casi ambigui, contraddittori e volutamente ingannevoli. Il sistema deve essere deterministico dove possibile, facilmente estendibile e corredato da test automatici. Non limitarti a descrivere l'architettura: produci gli artefatti funzionanti.

**Cosa si sta testando:** Costruire un sistema di triage con quattro artefatti verificabili: codice che compila, JSON valido, knowledge base e runbook. Il test delle CHIAMATE.

Divario OC/OF: **+479% tempo · +877% costo · +1.850% chiamate**

| OF                                           | OC (col marcatore)                            |
|----------------------------------------------|-----------------------------------------------|
| OF 136,8 s · 2 ch · \$0,01288 · 4/4 conformi | OC 792,6 s · 39 ch · \$0,12586 · 4/4 conformi |

Stessa conformita', 2 chiamate contro 39

**Consegnato:** OF 4 artefatti su 4 verificati — `ticket_triage.py` compila (19 funzioni), `test_cases.json` valido con 28 ticket, knowledge base 38 casi, runbook 18 sezioni. OC: 4 su 4 piu' tests/, schema e README (33 ticket, 26 casi, ~172 KB).

**Parere:** conformita' identica e oltre i minimi; OF vince sulle chiamate (2 contro 39) e sul contenuto della knowledge base (38 casi in un quarto del peso), **OC vince sull'aderenza letterale** (i test automatici richiesti, che OF ha omissso).

## Conclusioni

---

- **OF consegna sempre** nelle cinque prove di confronto (5 su 5) ed è la colonna più economica e rapida: **447,4 s e \$0,05684 totali** contro 1.623 s e \$0,25282 di OC.
- **Il divario è massimo dove OC ricostruisce da zero** (WEB1: +926% tempo) e minimo dove OC parte da artefatti esistenti: nei due run di SAAS1 e BUS1, 41 delle sue 44/22 chiamate erano riuso o revisione — un confronto non puro, e va detto.
- **La prova sulle chiamate (T7/TRIAGE1) è quella che pesa di più:** stessa conformità (4 artefatti su 4, oltre i minimi), **2 chiamate contro 39**. La knowledge base di OC pesa 4 volte quella di OF e contiene il 32% di casistiche in meno: il file grande non compra contenuto.