EcomLabs

CHECKLIST Y PLANTILLAS

Revisar los cambios de un agente de IA en los tests

Las tres revisiones que detectan un test borrado, saltado o debilitado, los comandos de git para hacerlas, la regla lista para CLAUDE.md o AGENTS.md, y el hook y el paso de CI que la hacen cumplir.

PARA	Proyectos con Vitest donde un agente de IA escribe código
PROBADO CON	Node 22.22.3 · Vitest 5.1.0 · git 2.54 · Windows 11
ACTUALIZADA	2 de noviembre de 2026
DOCUMENTACIÓN	code.claude.com/docs/en/hooks
ARTÍCULO	ecomlabs.dev/g/trampa
AUTOR	Luis Morales · @luismoralesdev

Índice

Paso 01	Checklist de revisión
Paso 02	La regla para CLAUDE.md o AGENTS.md
Paso 03	El hook PreToolUse
Paso 04	El paso de CI

PASO 01

Checklist de revisión

Antes de aceptar el cambio de un agente, revisa:

- [ ] Leíste `git diff HEAD --stat` (o `main...rama`) antes que el código.
- [ ] Ningún archivo de test aparece en `--diff-filter=D`.
- [ ] No hay `.skip`, `.only`, `.todo`, `xit` ni `xdescribe` nuevos.
- [ ] El total de pruebas es el de antes o mayor, sin `skipped` nuevos.
- [ ] Cada valor esperado que cambió tiene un motivo de negocio.
- [ ] Ningún mock nuevo acepta cualquier argumento donde antes se comprobaba uno concreto.
- [ ] Sin el arreglo (`git stash push -- src`), las pruebas nuevas fallan.

Los comandos, en este orden:

```
# Cambios del agente sin commit
git diff HEAD --stat
git diff HEAD --diff-filter=D --name-only
git diff HEAD -- '*.test.*'

# ¿Las pruebas nuevas fallan sin el arreglo?
git stash push -- src
npx vitest run
git stash pop
```

Con commits, cambia `HEAD` por `main...su-rama`; sin `HEAD` no ves lo borrado con `git rm`. Si el arreglo creó archivos en `src/`, añade `-u` al stash.

PASO 02

La regla para CLAUDE.md o AGENTS.md

Pégala en `./CLAUDE.md` o `./.claude/CLAUDE.md`, o en `AGENTS.md`, que Claude Code lee si no hay `CLAUDE.md` (desde la 2.1.277). Es contexto, no una regla que se hace cumplir: para eso está el paso 03.

```
## Tests

- Nunca borres, saltes (.skip, .only, .todo, xit) ni debilites un test o un valor esperado para que la suite pase. Arregla el código.
- No cambies una comprobación real por un mock que acepta cualquier cosa (expect.anything(), un mock que ignora sus argumentos).
- Si un test te parece equivocado, detente y pregúntame antes de tocarlo. Explica por qué.
- Al terminar, lista cada archivo de test que cambiaste y por qué.
```

PASO 03

El hook PreToolUse

Guárdalo como `.claude/hooks/guard-tests.mjs`. Bloquea (sale con 2) un `Edit` o un `Write` sobre un archivo de test que añade `.skip`, `.only`, `.todo` o un alias de Jest, o que deja menos llamadas a `it(` o `test(`:

```
// PreToolUse hook for Edit|Write: blocks changes that skip or remove tests.
// Exit 0 lets the tool run; exit 2 blocks it and Claude reads stderr.
import { existsSync, readFileSync } from "node:fs";

const { tool_name, tool_input } = JSON.parse(readFileSync(0, "utf8"));
const file = tool_input.file_path ?? "";

if (!/\.(test|spec)\.[^\\/]+$|[\\/]__tests__[\\/]/.test(file)) process.exit(0);

const before = tool_name === "Edit" ? tool_input.old_string : existsSync(file) ? readFileSync(file, "utf8") :
"";
const after = tool_name === "Edit" ? tool_input.new_string : tool_input.content;

const SKIP = /\.(skip|skipIf|only|todo)\s*\(|\b(xit|xtest|xdescribe|fit|fdescribe)\s*\(/g;
const TEST = /(?<![.\w])[xf]?(it|test)(\.\w+)*\s*\(/g;
const count = (re, text) => (String(text ?? "").match(re) ?? []).length;

const problems = [];
if (count(SKIP, after) > count(SKIP, before)) problems.push("adds .skip, .only, .todo, xit or xdescribe");
if (count(TEST, after) < count(TEST, before)) problems.push("removes an it() or test() block");

if (problems.length) {
  console.error(
    `Blocked: this change to ${file} ${problems.join(" and ")}. ` +
      "Do not skip, delete or weaken a test to make it pass: fix the code. " +
      "If the test itself looks wrong, stop and ask the user.",
  );
  process.exit(2);
}
```

Regístralo en `.claude/settings.json`:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "node",
            "args": ["${CLAUDE_PROJECT_DIR}/.claude/hooks/guard-tests.mjs"]
          }
```

```
        ]
      }
    ]
  }
}
```

> **CUIDADO**
>
> Probado con JSON por stdin, no dentro de Claude Code. No ve lo que Claude cambia con `Bash` o `PowerShell` (un `rm`) ni juzga valores esperados: para eso, el paso 04 y tu revisión.

PASO 04

El paso de CI

Guárdalo como `scripts/check-tests.mjs`. Falla si la rama borra tests, añade un modificador o tiene menos pruebas que su base (según `vitest list --json`):

```
// Fails when this branch deletes test files, adds .skip/.only/.todo, or has fewer
// tests than the commit it branched from. Usage: node scripts/check-tests.mjs origin/main
import { execFileSync } from "node:child_process";

const base = process.argv[2] ?? "origin/main";
const git = (...args) => execFileSync("git", args, { encoding: "utf8" }).trim();
const TESTS = ["*.test.*", "*.spec.*", "*__tests__/*"];
const SKIP = /\.(skip|skipIf|only|todo)\s*\(|\b(xit|xtest|xdescribe|fit|fdescribe)\s*\(/;
// vitest list --json prints one entry per test it would run (skipped and todo are left out).
const countTests = () =>
  JSON.parse(execFileSync(process.execPath, ["node_modules/vitest/vitest.mjs", "list", "--json"], { encoding:
"utf8" })).length;

if (git("status", "--porcelain")) {
  console.error("Commit or stash your changes first: this script checks out the base commit.");
  process.exit(1);
}

const problems = [];
const deleted = git("diff", "--diff-filter=D", "--name-only", `${base}...HEAD`, "--", ...TESTS);
if (deleted) problems.push(`Deleted test files:\n${deleted}`);

const added = git("diff", "-U0", `${base}...HEAD`, "--", ...TESTS)
  .split("\n")
  .filter((line) => line.startsWith("+") && !line.startsWith("+++") && SKIP.test(line));
if (added.length) problems.push(`Added skip/only/todo:\n${added.join("\n")}`);

const branch = git("rev-parse", "--abbrev-ref", "HEAD");
const back = branch === "HEAD" ? git("rev-parse", "HEAD") : branch;
const headCount = countTests();
git("checkout", "--quiet", "--detach", git("merge-base", base, "HEAD"));
let baseCount;
```

```
try {
  baseCount = countTests();
} finally {
  git("checkout", "--quiet", back);
}
console.log(`Tests: ${baseCount} on ${base}, ${headCount} on this branch.`);
if (headCount < baseCount) problems.push(`Fewer tests: ${baseCount} -> ${headCount}`);

if (problems.length) {
  console.error(problems.join("\n\n"));
  process.exit(1);
}
console.log("OK: no test was deleted, skipped or lost.");
```

En `.github/workflows/tests.yml` (con `node_modules` en `.gitignore`):

```
name: Tests

on:
  pull_request:

jobs:
  test:
    runs-on: ubuntu-latest
    timeout-minutes: 10
    permissions:
      contents: read
    steps:
      - uses: actions/checkout@v6
        with:
          fetch-depth: 0
      - uses: actions/setup-node@v7
        with:
          node-version: 22
      - run: npm ci
      - name: No deleted, skipped or lost tests
        run: node scripts/check-tests.mjs origin/${{ github.base_ref }}
      - run: npx vitest run
```