Resumo comparativo: Claude Code, OpenCode, IDE, CLI, MCP, AGENTS.md e CLAUDE.md

Visão geral

Claude Code e OpenCode pertencem à mesma classe geral de ferramentas: agentes de código com acesso a arquivos, terminal e fluxo de desenvolvimento, mas com filosofias diferentes.[cite:2][cite:114] Claude Code prioriza uma experiência mais gerenciada e centrada no ecossistema Anthropic, enquanto OpenCode prioriza abertura, liberdade de provedores e compatibilidade com convenções de outros agentes.[cite:69][cite:82][cite:114]

No fluxo discutido, o ponto principal foi entender como um ambiente já estruturado em Claude Code pode conviver com OpenCode sem perder memória de projeto, skills e instruções persistentes.[cite:84][cite:112] A resposta curta é que OpenCode foi desenhado para facilitar essa migração e consegue reaproveitar `CLAUDE.md` e `~/.claude/CLAUDE.md` quando `AGENTS.md` não existe.[cite:84][cite:89][cite:110][cite:112]

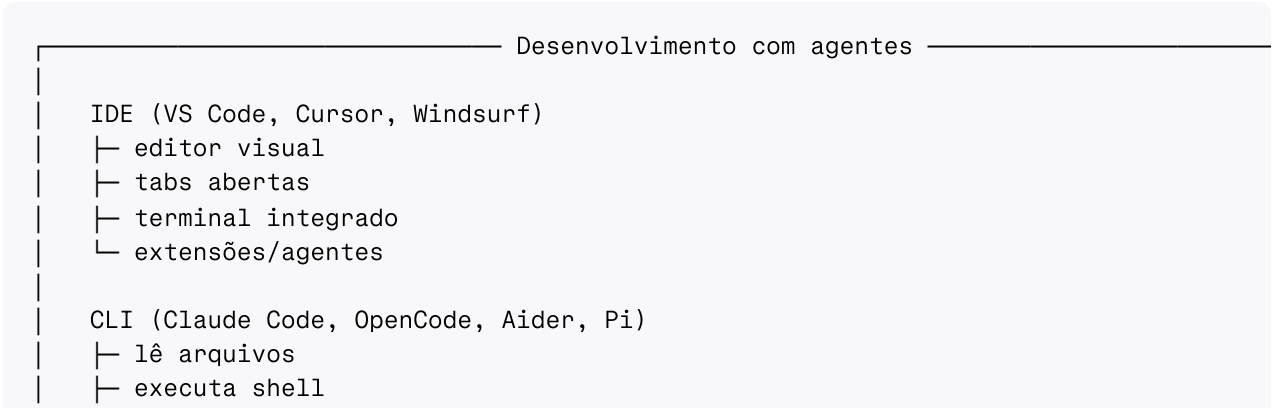
Conceitos-base

IDE, CLI e MCP

Uma **IDE** é um ambiente gráfico de desenvolvimento com editor, navegação, terminal integrado e contexto visual do projeto; exemplos típicos são VS Code, Cursor e Windsurf.[cite:29][cite:37] Já a **CLI** é um agente operando no terminal, com foco em delegação de tarefas, execução de comandos, leitura de arquivos e iteração baseada em saída de testes e shell.[cite:39][cite:42]

O **MCP (Model Context Protocol)** é um protocolo aberto para conectar agentes a ferramentas externas por meio de chamadas estruturadas, em vez de depender apenas de shell commands ou integrações ad hoc.[cite:31][cite:34][cite:40] Em geral, CLI é melhor para operações locais e ferramentas como `git`, `docker` e `kubectl`, enquanto MCP tende a fazer mais sentido para integrações externas, autenticação complexa e ecossistemas multiagente.[cite:33][cite:36]

Diagrama: onde cada peça entra



```
│   ├─ altera código
│   └─ valida por testes/logs
│
│   MCP
│   ├─ cliente de IA
│   ├─ tools/list
│   ├─ tools/call
│   └─ servidores externos (GitHub, Notion, APIs etc.)
│
└──────────────────────────────────────────────────────────────────────────────
```

Claude Code vs OpenCode

Diferenças estruturais

Claude Code é um agente oficial da Anthropic, com foco em qualidade de execução, experiência polida e integração em terminal, IDEs e outras superfícies do ecossistema da empresa.[cite:69][cite:73][cite:82] OpenCode é uma alternativa open source e model-agnostic, com suporte a dezenas de provedores, terminal/TUI, desktop app e extensão para VS Code e derivados.[cite:69][cite:70][cite:72][cite:118]

OpenCode também oferece extensão para VS Code e forks como Cursor, Windsurf e VSCodium, podendo ser iniciado a partir do terminal integrado com o comando `opencode`.[cite:68][cite:74][cite:76][cite:118] Isso significa que o uso do OpenCode não precisa ficar restrito ao terminal externo; ele pode conviver com o mesmo fluxo visual ao qual o usuário já estava habituado no Claude Code.[cite:68][cite:76][cite:118]

Quadro comparativo principal

Dimensão	Claude Code	OpenCode
Licença	Proprietário, oficial Anthropic [cite:69][cite:82]	MIT / open source [cite:69][cite:72]
Modelos	Claude apenas [cite:69][cite:82]	75+ provedores, incluindo Claude, GPT, Gemini e locais [cite:69][cite:72][cite:87]
Interface	Terminal + extensões oficiais + outras superfícies [cite:73][cite:82]	TUI, desktop app e extensão/integração com VS Code e forks [cite:70][cite:72][cite:76][cite:118]
Contexto do projeto	`CLAUDE.md` e camadas de memória do Claude [cite:95][cite:114]	`AGENTS.md` como padrão, com compatibilidade para `CLAUDE.md` [cite:84][cite:89][cite:110]
Extensibilidade	skills, MCP e ecossistema gerenciado [cite:73][cite:82]	`opencode.json`, plugins TypeScript, MCP e múltiplos providers [cite:54][cite:61][cite:114]
Custo da ferramenta	assinatura paga / ecossistema Anthropic [cite:69][cite:77][cite:82]	ferramenta gratuita; custo depende do provider/modelo escolhido [cite:69][cite:77][cite:82]

Visual: dois estilos de produto

```
Claude Code                          OpenCode
─────────────                        ────────────
Produto gerenciado                   Plataforma aberta
Modelo Anthropic-only                Multi-modelo
Fluxo mais polido                    Fluxo mais configurável
CLAUDE.md                            AGENTS.md (+ fallback CLAUDE.md)
Skills / MCP                         Plugins TS / MCP / config JSON
```

Como funcionam `CLAUDE.md`, `AGENTS.md` e compatibilidade

Claude Code lê `CLAUDE.md` como arquivo principal de contexto do projeto e não lê `AGENTS.md` diretamente.[cite:96][cite:99][cite:101][cite:105] Já OpenCode lê `AGENTS.md` como padrão e, para facilitar a migração, aceita `CLAUDE.md` como fallback quando `AGENTS.md` não existe no projeto.[cite:84][cite:89][cite:110][cite:112]

A ordem de resolução no OpenCode foi um dos pontos mais importantes da conversa: ele procura arquivos locais (`AGENTS.md`, `CLAUDE.md`), depois o arquivo global em `~/.config/opencode/AGENTS.md`, e por fim o `~/.claude/CLAUDE.md`, a menos que a compatibilidade com Claude Code seja desabilitada.[cite:84][cite:89][cite:112] O primeiro arquivo correspondente vence; portanto, se `AGENTS.md` e `CLAUDE.md` coexistirem, OpenCode usa `AGENTS.md` e ignora `CLAUDE.md` no nível local.[cite:84][cite:89][cite:112]

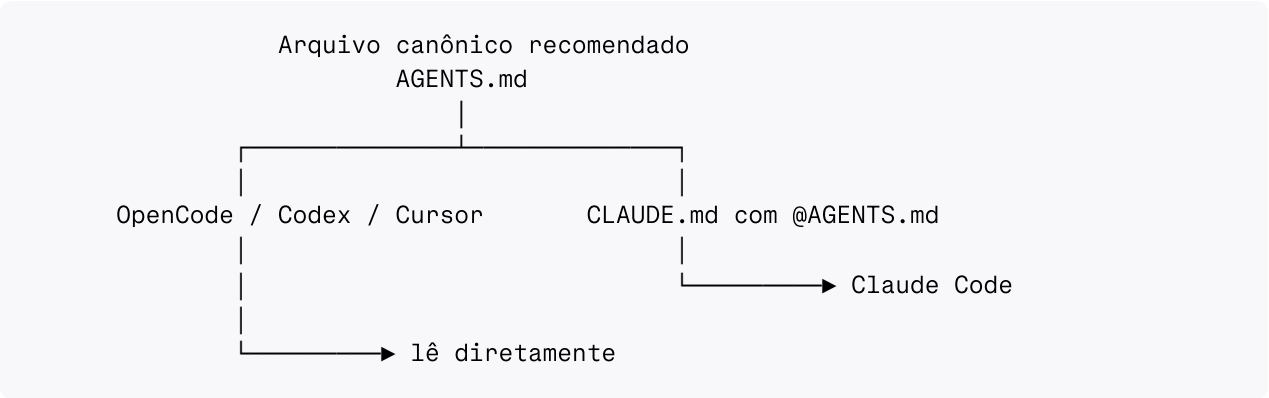
Matriz de leitura de arquivos

Arquivo	Claude Code	OpenCode
`CLAUDE.md` no projeto	Lê nativamente [cite:95][cite:101]	Lê se `AGENTS.md` não existir [cite:84][cite:89][cite:112]
`AGENTS.md` no projeto	Não lê diretamente [cite:96][cite:99][cite:101]	Lê nativamente [cite:84][cite:89]
`~/.claude/CLAUDE.md`	Lê como camada global [cite:95][cite:105]	Lê como fallback global de compatibilidade [cite:84][cite:89][cite:112]
`~/.config/opencode/AGENTS.md`	Não usa [cite:101]	Lê como regra global prioritária do OpenCode [cite:54][cite:61][cite:84]

Melhor arranjo para usar os dois

Para usar Claude Code e OpenCode no mesmo repositório sem duplicação manual, a estratégia mais estável é manter `AGENTS.md` como arquivo canônico e criar um `CLAUDE.md` mínimo com `@AGENTS.md` no topo, ou usar um symlink quando o ambiente permitir.[cite:96][cite:98][cite:100][cite:101][cite:105] Isso faz Claude Code continuar lendo `CLAUDE.md`, enquanto OpenCode e outros agentes leem `AGENTS.md` diretamente.[cite:98][cite:100][cite:101][cite:105]

Diagrama: compatibilidade de arquivos



Diferentes modelos dentro do OpenCode

Quando OpenCode é usado com diferentes modelos — como Claude, GPT, DeepSeek, GLM, MiniMax, Qwen ou outros compatíveis — quem controla a sessão é o **harness do OpenCode**, não o modelo individual.[cite:83][cite:86][cite:87][cite:91] Isso significa que o processo de localizar arquivos de instrução, injetar o contexto no prompt, apresentar ferramentas e controlar permissões continua sendo do OpenCode; o modelo entra principalmente na parte de raciocínio e geração da resposta.[cite:83][cite:84][cite:89]

Na prática, se o projeto foi originalmente criado no Claude Code e usa `CLAUDE.md`, OpenCode continuará lendo esse arquivo na ausência de `AGENTS.md`, independentemente do modelo ativo na sessão.[cite:83][cite:84][cite:89][cite:110] Todos esses modelos seguem as instruções que o harness fornecer, mas a qualidade com que obedecem, priorizam convenções e mantêm coerência varia conforme a capacidade do modelo escolhido.[cite:83][cite:87][cite:91]

Modelo vs harness

Camada	Quem manda	Exemplo
Descoberta de contexto	Harness do OpenCode [cite:84][cite:89]	localizar `AGENTS.md` ou `CLAUDE.md`
Injeção de instruções	Harness do OpenCode [cite:83][cite:84]	incluir regras do projeto no prompt
Ferramentas e permissões	Harness do OpenCode [cite:54][cite:61][cite:114]	bash, leitura de arquivo, plugins, MCP
Qualidade do raciocínio	Modelo escolhido [cite:83][cite:87][cite:91]	quão bem interpreta e executa
Estilo e aderência	Modelo escolhido, dentro dos limites do harness [cite:83][cite:87]	seguir padrões, escrever testes, explicar melhor

O que são plugins TypeScript e `opencode.json`

Uma diferença relevante é que OpenCode oferece um sistema de configuração estruturado em `opencode.json` e suporta plugins em TypeScript ou JavaScript para estender o comportamento do agente.[cite:54][cite:56][cite:61] Esses arquivos podem definir modelos, providers, permissões, MCP servers, agentes customizados, comandos e diretórios de plugins, tanto no nível global quanto no nível do projeto.[cite:54][cite:59][cite:61][cite:64]

Isso contrasta com o fluxo mais familiar do Claude Code, em que o ponto de customização do usuário gira mais em torno de `CLAUDE.md`, skills, MCP e convenções próprias do ecossistema Claude.[cite:95][cite:114] Em OpenCode, a configuração é mais "programável": além do texto de instruções, há uma camada formal de configuração e automação baseada em JSON/JSONC e plugins carregados de diretórios como `.opencode/plugins/` ou `~/.config/opencode/plugins/`. [cite:56][cite:59][cite:61]

Estrutura típica do OpenCode

```
~/.config/opencode/
├── opencode.json          # config global
├── AGENTS.md              # regras globais do OpenCode
└── plugins/               # plugins TS/JS

projeto/
├── AGENTS.md              # regras do projeto (preferido)
├── CLAUDE.md              # fallback/compatibilidade
├── opencode.json          # config do projeto
└── .opencode/
    ├── agents/
    ├── command/
    └── plugins/
```

Tabela: `CLAUDE.md` vs `opencode.json`

Aspecto	`CLAUDE.md`	`opencode.json`
Formato	Markdown textual [cite:95][cite:101]	JSON ou JSONC [cite:59][cite:61]
Objetivo	instruções e convenções do projeto [cite:95][cite:114]	configuração estrutural de runtime, modelos, permissões, MCP e plugins [cite:54][cite:61][cite:64]
Ferramenta principal	Claude Code [cite:95][cite:101]	OpenCode [cite:54][cite:61]
Portabilidade	alta entre ferramentas que aceitam markdown compatível [cite:84][cite:110]	específica do ecossistema OpenCode [cite:54][cite:61]
Programabilidade	baixa / declarativa [cite:95][cite:114]	alta / estruturada e extensível [cite:56][cite:61]

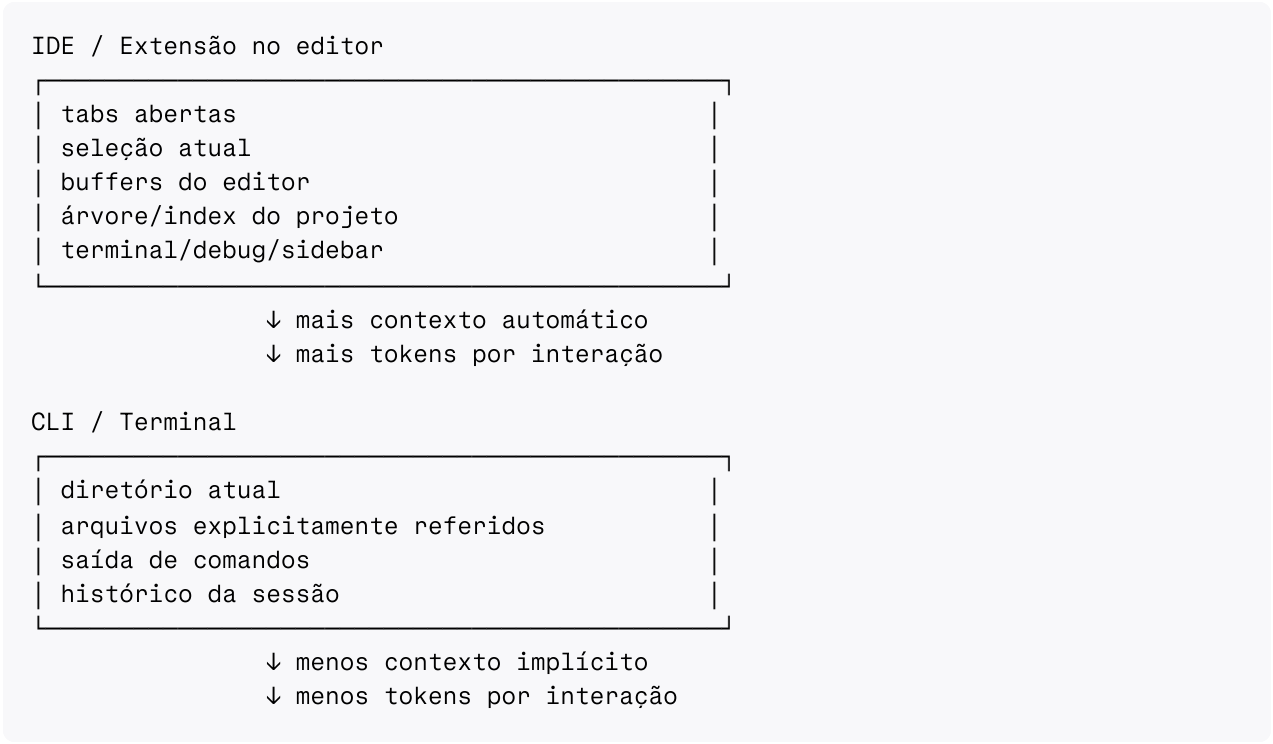
Por que IDE costuma gastar mais tokens que CLI

A conversa também diferenciou o custo de contexto entre IDE e CLI. Em linhas gerais, ferramentas acopladas à IDE tendem a carregar mais contexto implícito, como arquivos abertos, seleção de código, índice do projeto e estado do editor, enquanto ferramentas de terminal tendem a receber contexto mais explícito e controlado.[cite:29][cite:39][cite:40] [cite:42]

Algumas análises e relatos comparativos apontam que fluxos CLI podem ser substancialmente mais eficientes em tokens que integrações mais pesadas, especialmente quando há muitas ferramentas, painéis e schemas disponíveis simultaneamente.[cite:30][cite:36][cite:39] Isso não

significa que IDE seja “pior”; significa apenas que ela compra conveniência e contexto automático ao custo de mais volume de informação por interação.[cite:29][cite:39][cite:40]

Visual: custo de contexto



Recomendações práticas

Se o fluxo principal é Claude Code no VS Code

Esse fluxo continua fazendo sentido quando a prioridade é ergonomia visual, histórico de chat integrado, navegação por arquivos e continuidade da experiência já dominada.[cite:49][cite:50][cite:73] Nesse caso, vale manter `CLAUDE.md`, skills e hábitos existentes como base operacional principal.[cite:95][cite:114]

Se a meta é ganhar liberdade de modelo

OpenCode passa a ser útil como camada complementar quando a necessidade é alternar entre Claude, GPT, Gemini, DeepSeek ou modelos locais sem reescrever toda a memória de projeto.[cite:69][cite:72][cite:83][cite:91] Nesse cenário, o caminho mais seguro é manter compatibilidade de arquivos e testar cada família de modelo em tarefas específicas, observando custo, consistência e aderência às instruções.[cite:83][cite:87][cite:91]

Arranjo recomendado

Objetivo	Estratégia sugerida
Continuar produtivo no fluxo atual	manter Claude Code no VS Code como ferramenta principal [cite:49][cite:73]

Objetivo	Estratégia sugerida
Ganhar portabilidade entre agentes	criar `AGENTS.md` como fonte canônica e `CLAUDE.md` com `@AGENTS.md` [cite:96][cite:100][cite:101]
Reduzir lock-in de modelo	usar OpenCode para alternar providers e modelos [cite:69][cite:72][cite:83]
Evitar perda de contexto de projeto	preservar markdowns de instrução e regras globais/projeto [cite:84][cite:89][cite:110]
Fazer customizações avançadas	explorar `opencode.json` e plugins TypeScript no OpenCode [cite:54][cite:56][cite:61]

Síntese final

A principal conclusão é que Claude Code e OpenCode não são concorrentes mutuamente excludentes; eles podem compor um fluxo híbrido bastante eficiente.[cite:69][cite:71][cite:79] Claude Code continua forte como ambiente gerenciado e familiar, enquanto OpenCode amplia flexibilidade de modelos, abertura e possibilidades de customização sem exigir abandono imediato da estrutura já construída em `CLAUDE.md`.[cite:69][cite:82][cite:84][cite:110]

Para quem já investiu tempo em refinar memória global, regras por projeto e skills no Claude, a forma mais prática de evoluir é preservar essa base, introduzir `AGENTS.md` de forma compatível e usar OpenCode como uma segunda camada estratégica de execução e experimentação.[cite:84][cite:89][cite:96][cite:101] Essa abordagem reduz lock-in, melhora portabilidade e mantém o melhor dos dois mundos: o conforto operacional do Claude e a flexibilidade arquitetural do OpenCode.[cite:69][cite:72][cite:79][cite:114]