USING CODE AGENTS

Siempre has escrito el codigo. Y si el codigo pudiera escribirse a si mismo?

Fran Dieguez

La antiguo matrix del dev

// el programador atrapado

¿Te suena esto?

- Refactors interminables de componentes
- Escribir tests repetitivos una y otra vez
- Debugging que consume horas
- Scaffolding de proyectos desde cero
- Tareas mecánicas que no aportan valor
- Copy-paste de patrones conocidos

El 40% del código que escribes hoy podría ser generado y ejecutado por un agente de forma autónoma.

La nueva realidad del desarrollo

// el cambio cuantico

IAs como asistentes

- Sugiere código
- Tú decides y ejecutas
- Reactivo a tu cursor
- Sin acceso al sistema
- Sin loop de corrección

→

IAs como Agentes

- Planifica y ejecuta
- Accede al sistema de ficheros
- **Ejecuta** comandos shell
- Verifica y corrige errores
- Loop autónomo hasta completar

Wake up Neo...

Que es un code agent

// Neo se despierta

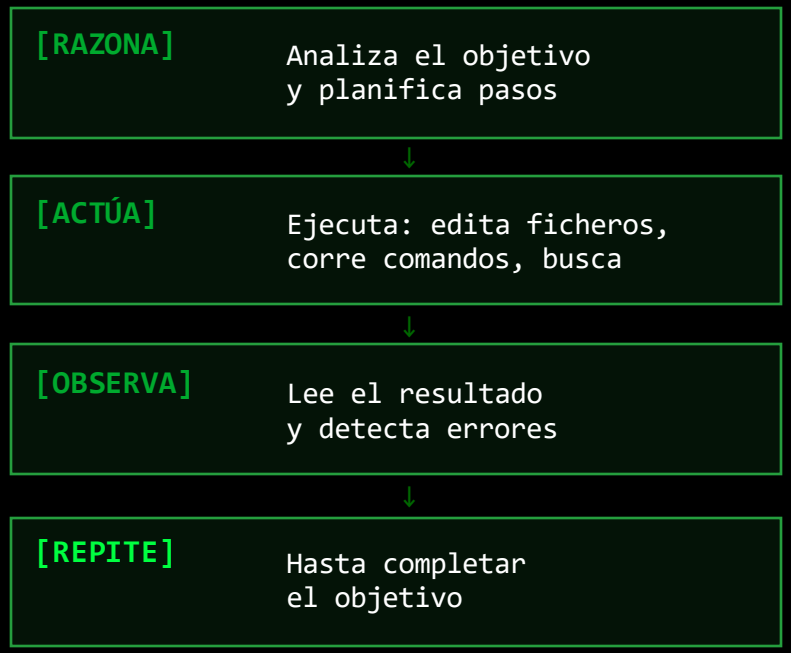
Neo abre los ojos por primera vez. No sabe nada.

El agente en su estado base: existe, razona, pero no puede hacer nada en tu mundo.

Cómo funciona internamente

// La cápsula de entrenamiento

El loop ReAct



Como se organiza?

- Sigue el patrón **ReAct + Divide y venceras**

Como piensa?

- Memoria (aka contexto): historial, codebase, resultados anteriores, …
- Cerebro (aka modelos): openai, claude, zen, qwen

Como actua?

- Heramientas internas
  - files I/O: acceso a ficheros
  - shell: ejecutar comandos
  - glob: buscar ficheros
  - diff: aplicar parches
- Herramientas externas
  - Plugins de browser
  - Servidores MCP

El equipo


Trinity
// la ejecutora
Tank
// el operador
Switch y Apoc
// soldados apoyo
Cypher
// el riesgo
Neo
// ?
Mouse
// el freak
Dozer
// el piloto
Morpheus
// el lider
THE MATRIX

"I know Kung Fu" – Las Skills

Operator _ Cómo gestionar las Skills

// La cápsula de entrenamiento

- Herramientas plug & play que amplían lo que el agente puede hacer
  - + Se definen típicamente como scripts o comandos que el agente puede invocar con parámetros
  - + **Ejemplos prácticos:** kotlin, spring boot, design systems, frontend dev
- **Repositorio centralizado:** skills.sh
- **Gestión:**
  - + se registran en la configuración del proyecto o globales
  - + se pueden versionar en el repo
  - + y compartir con el equipo

SKILLS

THE OPEN AGENT SKILLS ECOSYSTEM

Enchufarse a Matrix

// Neo ya está listo. Se enchufa.

- **Claude Code** (Anthropic): CLI potente, muy bueno en razonamiento y contexto largo
- **Codex / ChatGPT** (OpenAI): integrado en ecosistema OpenAI
- **Cursor**: IDE basado en VSCode con agente integrado
- **Aider**: CLI open source, enfocado en git-aware coding
- **OpenCode**: alternativa open source, multimodelo, basada en terminal.

El mercado está fragmentado pero convergiendo hacia agentes con acceso total al sistema

opencode

- https://opencode.ai/
- Proyecto open source escrito en **Typescript** con interfaz TUI (terminal UI)
- Soporta múltiples proveedores:
  - **OpenAI, Anthropic, Google Gemini, y modelos locales vía Ollama**
  - No te ata a un proveedor concreto — puedes cambiar el modelo según el caso de uso o coste
- Al ser terminal-first encaja perfectamente en flujos de trabajo dev sin salir del entorno
- Configuración mediante ficheros JSON, fácil de versionar y compartir en equipo


Demo time
Follow the white rabbit.

Seguridad

// En Matrix también hay sentinels

⚠ API Keys y secretos

Nunca hardcodeadas. Usar variables de entorno + .gitignore.
Gestores como envx, Vault, Doppler o AWS Secrets.

⚠ Permisos mínimos

El agente solo accede a lo que necesita. Sin credenciales de admin ni acceso directo a producción.

⚠ Revisión de código

Código generado puede introducir vulnerabilidades sutiles. Siempre revisar antes de mergear.

⚠ Sandboxing

Ejecutar el agente en Docker o VMs aisladas. Limitar el blast radius si algo sale mal.
Modelos locales

⚠ Skills de terceros

Tratar como cualquier dependencia externa. Revisar el código. No instalar de fuentes no confiables.

⚠ Logs y contexto

Los logs pueden contener datos sensibles. Gestionar dónde se almacenan y quién tiene acceso.

Buenas practicas

- Plugins recomendados: opencode-agent-memory, opencode-agent-skills, envsitter-guard
- AGENTS.md - agents.md Ej. map-viewer
- Skills.sh
  - **Superpowers** – https://github.com/obra/superpowers Turns the agent into a senior developer
  + **GSD (Get Shit Done)** – https://github.com/glittercowboy/get-shit-done It interviews you about your idea, builds a full spec, then executes phase by phase with verification.
  + **Awesome Claude Code** – https://github.com/hesreallyhim/awesome-claude-code The ultimate directory of Claude Code skills, hooks, slash commands, plugins, and tools
  + **UI/UX Pro Max Skill** – https://github.com/nextlevelbuilder/ui-ux-pro-max-skill Auto-generates a complete design system so your app looks like a designer touched it instead of AI-generated slop.
  - **Obsidian Skills** – https://github.com/kepano/obsidian-skills Second brain on autopilot.
- Envx: https://github.com/solvingj/envx

¿Hasta dónde llega el rabbit hole?

Conclusiones

> Los agents no reemplazan al dev, multiplican su productividad
> Seguridad y control humano son esenciales en prod
> OpenCode = control total sin vendor lock-in
> Las Skills son el multiplicador real del agente
> Futuro: memoria persistente, agentes especializados

"

No te preguntamos
si confías en la IA.

Te preguntamos
si sabes cómo controlarla.

— The Matrix, 2199

[TOMA LA PASTILLA VERDE. EMPIEZA HOY.]