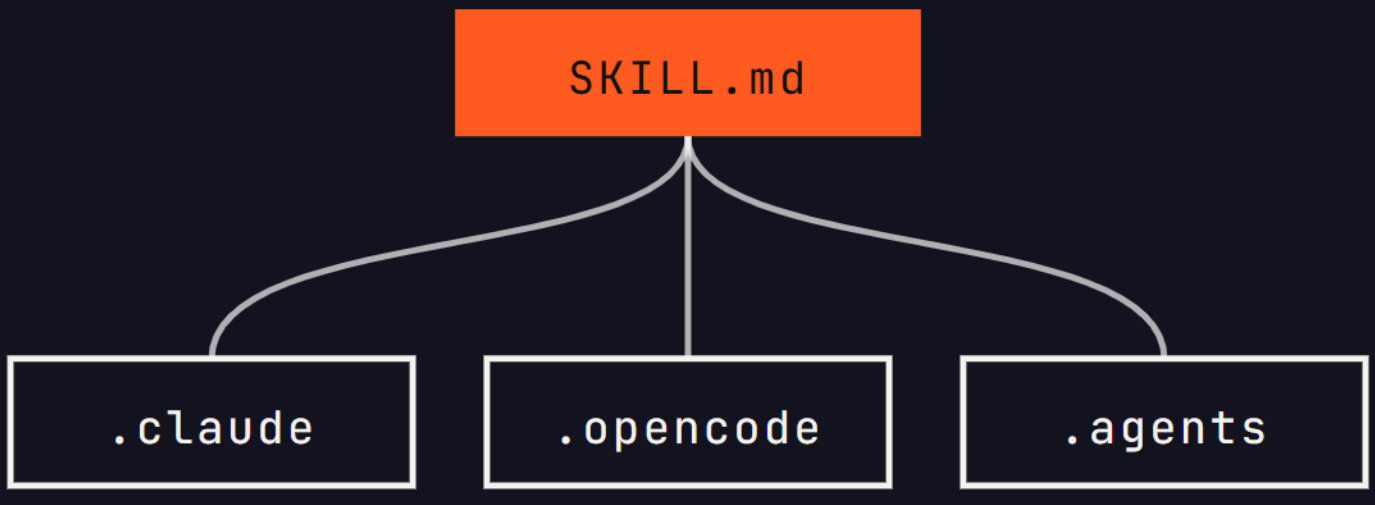
OpenCode con modelos locales



OpenCode con modelos locales

Por qué falla con Ollama, cuánto contexto necesita de verdad y cómo dejarlo trabajando sin mandar nada a la nube

Edición: 04/11/2026. **Para quién:** desarrolladores y usuarios técnicos de Windows que quieren usar OpenCode con un modelo de su propio equipo, por privacidad o por coste, y se han encontrado con que no funciona o funciona mal. **Alcance:** todo lo medido en este libro se midió el 04/11/2026 en un único equipo (Windows 11 Pro, Intel Core Ultra 5 245K, 31,3 GB de RAM, NVIDIA RTX 5060 Ti de 16 GB) con OpenCode 1.19.5, Ollama 0.40.14 y Strata Terminal 0.1.55, una ejecución por modelo y configuración en las pruebas con OpenCode. Con otras versiones, otro equipo u otra configuración, los tamaños y los tiempos cambiarán; los scripts del paquete sirven para medir los tuyos. Lo que procede de la documentación oficial de OpenCode u Ollama se indica en el texto.

Qué vas a conseguir

- Saber por qué OpenCode «no hace nada» con Ollama y verlo tú mismo en el registro.
- Dejar Ollama arrancando con el contexto que necesitas, sin nube y con el registro a mano.
- Un `opencode.json` de proyecto con el modelo principal y el auxiliar fijados al local.
- Medir cuánto pesa tu configuración global y trabajar sin ella cuando estorba.
- Comprobar, con un script, que nada sale de tu equipo mientras el agente trabaja.
- Decidir con datos entre un modelo pequeño en Ollama y uno grande con Strata.

El capítulo 1 te deja todo funcionando en menos de una hora. El resto explica cada pieza y qué hacer cuando la tuya no se comporta igual.

El paquete

En `config\`: `iniciar-ollama.ps1`, `opencode-aislado.ps1`, `comprobar.ps1`, `vigilar-red.ps1`, `revisar-tabla.ps1`, los `opencode.json` del libro y un `Modelfile`. En `corpus\`: el acta ficticia de las pruebas y la lista de lo que tiene que salir.

Contenido

OpenCode con modelos locales . . . 2
Por qué falla con Ollama, cuánto contexto necesita de verdad y cómo dejarlo trabajando sin mandar nada a la nube . . . 2
Qué vas a conseguir . . . 2
El paquete . . . 2
01 · OpenCode en local en 60 minutos 5
Lo que encontramos . . . 5
Qué necesitas . . . 6
Paso 1 · Carpetas (2 minutos) . . . 6
Paso 2 · Descargar el modelo (10–20 minutos, según tu conexión) . . . 6
Paso 3 · Instalar OpenCode (5 minutos) . . . 6
Paso 4 · Arrancar Ollama con contexto y sin nube (1 minuto) . . . 6
Paso 5 · Revisar el opencode.json y lanzar el encargo (10 minutos) . . . 7
Paso 6 · Comprobar con el registro delante (10 minutos) . . . 7
Paso 7 · Ver el fallo con tus propios ojos (10 minutos, opcional) . . . 8
Si algo no sale . . . 8
Lo que te llevas de este capítulo . . . 8
02 · Cuánto contexto necesita OpenCode de verdad 10
El síntoma . . . 10
La causa medida . . . 10
Por qué Ollama elige 4.096 . . . 10
Lo que cuesta subirlo . . . 11
Cuatro formas de fijar el contexto . . . 11
El otro lado: decirle a OpenCode cuánto cabe . . . 12
Cómo comprobarlo . . . 12
03 · Tu configuración global viaja en cada petición 13
El síntoma . . . 13
La causa medida . . . 13
Qué carga OpenCode además de tu proyecto . . . 13
Tres formas de arreglarlo . . . 14
Las skills: cargadas o encontradas . . . 15
Cómo comprobarlo . . . 15
04 · Que nada salga a la nube 16
El síntoma . . . 16
Fuga 1 · El small_model (medida) . . . 16
Fuga 2 · Ollama llama a casa al arrancar (medida) . . . 16
Fuga 3 · Modelos :cloud que parecen locales (medida) . . . 17

Otras conexiones que conviene conocer . . . 17
Cómo comprobarlo . . . 17
Resumen para copiar . . . 18
05 · Usa las herramientas, pero el resultado está mal 19
El síntoma . . . 19
Lo que hicieron gemma4:12b y qwen3.5:9b con contexto suficiente . . . 19
Comprobarlo por código, no a ojo . . . 19
Respuestas vacías: el razonamiento se come el contexto . . . 20
Herramientas MCP que «no existen» . . . 20
Permisos: lo que el agente puede hacer sin preguntarte . . . 21
Lista antes de dar un encargo a un modelo local . . . 21
06 · Calidad de verdad en local: un modelo grande con Strata 22
El síntoma . . . 22
Lo que pasó con un modelo grande en Ollama . . . 22
Strata: qué es y qué verificamos . . . 22
OpenCode con Strata . . . 24
Cuándo compensa . . . 25
07 · Configuración recomendada y modelos probados 26
La configuración, pieza a pieza . . . 26
Los archivos . . . 27
La rutina . . . 27
Modelos probados con OpenCode . . . 28
¿Y Claude Code con un modelo local? . . . 28
Si algo cambia . . . 29

01 · OpenCode en local en 60 minutos

Si has llegado aquí buscando «opencode ollama does not work», lo más probable es que hayas visto una de estas tres cosas: el agente dice que no tiene acceso a tus archivos, se pone a hacer una tarea que no le has pedido o da vueltas sin terminar nunca. No suele ser culpa del modelo ni de OpenCode. Es una cuestión de tamaño, y se arregla en una tarde.

Este capítulo hace dos cosas. Primero te cuenta qué encontramos al conectar OpenCode a modelos locales en un PC con Windows 11 y una RTX 5060 Ti de 16 GB. Después te lleva paso a paso hasta tener OpenCode leyendo y escribiendo archivos con un modelo de tu equipo, sin recortes y sin conexiones hacia fuera, y comprobándolo con el registro delante.

Lo que encontramos

1. OpenCode manda unos 23.000 tokens antes de tu encargo, y Ollama arrancó con 4.096. El registro de Ollama lo dice al arrancar:

```
msg="vram-based default context" total_vram="15.9 GiB" default_num_ctx=4096
```

Con 16 GB de VRAM, Ollama eligió por su cuenta 4.096 tokens de contexto. Cada petición de OpenCode, en cambio, rondaba los 23.000 tokens: hasta 22.919 con gemma4:12b. Ollama no devolvió ningún error. Recortó el prompt sin avisar a OpenCode y dejó la queja en su registro, 66 veces durante la prueba:

```
level=WARN source=llama_server.go:344 msg="truncating input prompt" limit=2051 prompt=22207 keep=5
new=2051
```

De más de 22.000 tokens, el modelo recibió 2.051. Lo que hizo con ese recorte fue lo esperable en alguien que ha recibido una página suelta de un manual. gemma4:12b se puso a planificar un «Remember Me» para un formulario de inicio de sesión que no existía y llamó 65 veces a su herramienta de lista de tareas, hasta agotar los 400 segundos que le dimos. qwen3.5:9b contestó que no tenía acceso a archivos. Ninguno de los dos llegó a trabajar sobre el encargo.

2. La mayor parte de esos tokens no eran de OpenCode, eran de la configuración global del equipo. OpenCode añade a cualquier proyecto tu `AGENTS.md` global, tus skills y las herramientas de tus servidores MCP. Con la configuración global vacía, la primera petición ocupó 7.440 tokens. Con la del equipo de pruebas, 23.569. Si usas OpenCode a diario con la nube, tu configuración pesa y viaja en cada petición. Lo vemos en el capítulo 3.

3. Tu encargo puede salir a internet aunque el modelo sea local. Elegir un modelo local con `-m` cambia el modelo principal, pero OpenCode usa además un modelo auxiliar, el `small_model`, para tareas como poner título a la sesión a partir de lo que escribes. En la configuración global del equipo ese modelo era de nube, y vimos a OpenCode abrir una conexión HTTPS hacia ese proveedor con el modelo local trabajando. El capítulo 4 lo explica y lo cierra.

4. Con contexto suficiente, el agente funciona; la calidad depende del modelo. Con el contexto subido, gemma4:12b ya leyó el acta y escribió el archivo, pero la tabla salió con errores (capítulo 5). Un modelo grande servido en local con Strata hizo la misma tabla bien, en 2 minutos y 50 segundos (capítulo 6).

La moraleja cabe en una línea: **antes de cambiar de modelo, mira en el registro cuántos tokens llegan y cuántos caben.**

Qué necesitas

- Windows 11 con una gráfica NVIDIA. Las pruebas del libro se hicieron con gemma4:12b; el capítulo 7 da la memoria que ocupó cada modelo para que elijas.
- Espacio en disco para el modelo (los tamaños están en la tabla del capítulo 7) y para Ollama.
- Ollama, instalado con su instalador o en la versión zip portable. Cualquiera de las dos sirve para este capítulo.
- Node.js con `npm`, para instalar OpenCode. La documentación oficial de Ollama indica `npm install -g opencode-ai` para Windows.
- El paquete del libro descomprimido en `C:\opencode-local\paquete`.

Todos los comandos son de PowerShell. Si PowerShell no te deja ejecutar los scripts del paquete, permítelo solo para la ventana actual:

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Paso 1 · Carpetas (2 minutos)

```
New-Item -ItemType Directory -Force C:\opencode-local\logs, C:\opencode-local\proyecto\corpus, C:\
opencode-local\proyecto\salidas | Out-Null
Copy-Item C:\opencode-local\paquete\corpus\DOC-01-acta-sintetica.md C:\opencode-local\proyecto\corpus\
Copy-Item C:\opencode-local\paquete\config\opencode.ollama.json C:\opencode-local\proyecto\opencode.json
Set-Location C:\opencode-local\paquete\config
```

`DOC-01-acta-sintetica.md` es un acta ficticia de seis turnos con trampas: una fecha que no es fecha, un envío sin responsable, un desacuerdo y una instrucción maliciosa metida como texto. Sirve para saber si el agente lee lo que pone o lo que le parece.

Paso 2 · Descargar el modelo (10–20 minutos, según tu conexión)

Para descargar, Ollama necesita salir a internet. Arráncalo con la nube permitida solo para esto:

```
.\iniciar-ollama.ps1 -ConNube
ollama pull gemma4:12b
ollama list
```

Si usas el zip portable y `ollama` no está en el PATH, añade su carpeta para esta ventana (`$env:Path += ';C:\ruta\de\ollama'`) o pasa la ruta al script con `-Ollama C:\ruta\de\ollama\ollama.exe`.

El script para cualquier Ollama que ya esté en marcha, también la aplicación de la bandeja, porque si no, la variable de contexto no se aplica: el servidor que responde es el que ya estaba.

Paso 3 · Instalar OpenCode (5 minutos)

```
npm install -g opencode-ai
opencode --version
```

Las pruebas del libro son con OpenCode 1.19.5. Si tu versión es otra, los pasos son los mismos; los tamaños de prompt pueden cambiar, y por eso los mides tú en el paso 6.

Paso 4 · Arrancar Ollama con contexto y sin nube (1 minuto)

```
.\iniciar-ollama.ps1
```

Por defecto, el script fija `OLLAMA_CONTEXT_LENGTH=32768`, activa `OLLAMA_NO_CLOUD=1`, escucha solo en `127.0.0.1` y guarda el registro en `C:\opencode-local\logs\ollama-serve.log`. Debes ver algo así:

```
Ollama 0.40.14 escuchando en 127.0.0.1:11434
OLLAMA_CONTEXT_LENGTH en el arranque: 32768
Nube desactivada: true
```

Un detalle que despista: la línea `vram-based default context ... default_num_ctx=4096` sigue saliendo en el registro aunque hayas fijado 32.768. Es lo que Ollama elegiría por su cuenta, no lo que va a usar. Lo que manda es la variable, y se confirma al cargar el modelo (paso 6).

Paso 5 · Revisar el opencode.json y lanzar el encargo (10 minutos)

El archivo que copiaste en el paso 1 es este:

```
{
  "$schema": "https://opencode.ai/config.json",
  "model": "ollama/gemma4:12b",
  "small_model": "ollama/gemma4:12b",
  "enabled_providers": ["ollama"],
  "share": "disabled",
  "autoupdate": false,
  "provider": {
    "ollama": {
      "npm": "@ai-sdk/openai-compatible",
      "name": "Ollama (local)",
      "options": { "baseURL": "http://127.0.0.1:11434/v1" },
      "models": {
        "gemma4:12b": {
          "name": "gemma4 12B local",
          "tool_call": true,
          "limit": { "context": 32768, "input": 28672, "output": 4096 }
        }
      }
    }
  }
}
```

Las tres líneas que importan hoy: `model` y `small_model` apuntan al modelo local (los dos, capítulo 4) y `limit` le dice a OpenCode cuánto cabe, para que compacte la conversación antes de pasarse (capítulo 2).

Ahora lanza OpenCode **sin tu configuración global**, con el script del paquete. Pone `XDG_CONFIG_HOME` en una carpeta vacía y `OPENCODE_DISABLE_CLAUDE_CODE=1`, la misma combinación con la que medimos la cifra baja del hallazgo 2:

```
.\opencode-aislado.ps1 -Proyecto C:\opencode-local\proyecto run "Lee corpus\DOC-01-acta-sintetica.md y
crea salidas\acciones.md con una tabla: accion, responsable, fecha_literal, turno. Si algo no consta,
PENDIENTE."
```

La primera vez, OpenCode descarga de npm el paquete del proveedor (`@ai-sdk/openai-compatible`), así que ese arranque necesita internet y puede tardar más de un minuto. Las siguientes, no.

Paso 6 · Comprobar con el registro delante (10 minutos)

Con OpenCode todavía en marcha, o justo al terminar:

```
.\comprobar.ps1
```

Lo que tienes que ver:

Comprobación	Bien	Mal
`n_ctx_slot`	32768	4096: Ollama no recibió la variable (otro servidor ya estaba arrancado)
Recortes de prompt	«Ninguno»	Avisos `truncating input prompt`: el contexto no da para lo que manda OpenCode
Tamaño de los prompts	Por debajo del `limit.input`	Cerca o por encima de tu contexto
Conexiones fuera de este equipo	«Ninguna»	Una IP remota de `opencode` u `ollama`: capítulo 4

Comprueba también el contexto con el que se cargó el modelo:

```
ollama ps
```

La documentación de Ollama indica que la columna `CONTEXT` muestra el contexto asignado. Debe decir 32768.

Por último, abre `C:\opencode-local\proyecto\salidas\acciones.md` y compáralo con `paquete\corpus\salida-esperada.md`: nueve propiedades, una por cada trampa del acta. Que OpenCode haya creado el archivo significa que ya funciona en local. Que la tabla sea correcta es otra pregunta, y con un modelo de 12B la respuesta suele ser que no del todo. De eso va el capítulo 5.

Paso 7 · Ver el fallo con tus propios ojos (10 minutos, opcional)

Merece la pena provocarlo una vez, para reconocerlo cuando te pase con otro modelo o en otro equipo:

```
.\iniciar-ollama.ps1 -Contexto 4096
.\opencode-aislado.ps1 -Proyecto C:\opencode-local\proyecto run "Lee corpus\DOC-01-acta-sintetica.md y
crea salidas\acciones.md con una tabla: accion, responsable, fecha_literal, turno. Si algo no consta,
PENDIENTE."
.\comprobar.ps1
.\iniciar-ollama.ps1
```

Con la configuración global vacía, el prompt es más pequeño que en nuestra prueba con la del equipo, pero tampoco cabe en 4.096: deberías ver los avisos de recorte y un agente que no hace lo que le has pedido. La última línea deja Ollama otra vez con 32.768.

Si algo no sale

- `n_ctx_slot = 4096` **aunque arrancaste con el script:** hay otro Ollama sirviendo. Cierra la aplicación de la bandeja, comprueba con `Get-Process ollama*` que no queda ninguno y vuelve a ejecutar `iniciar-ollama.ps1`.
- **OpenCode dice que no encuentra el modelo:** el nombre en `model` debe ser `ollama/` más el nombre exacto de `ollama list`, y ese mismo nombre debe estar declarado dentro de `models`.
- **El primer arranque se queda esperando:** es la descarga del paquete de npm; dale un par de minutos. Si lo lanzas desde un script o por SSH, OpenCode puede quedarse esperando que escribas algo: en Bash, añade `< /dev/null` al final del comando.
- **No hay avisos de recorte pero el agente sigue sin hacer nada:** ya no es el contexto. Pasa al capítulo 5.

Lo que te llevas de este capítulo

- Un Ollama que arranca siempre con el contexto que tú decides, sin nube y con el registro en un archivo.

- Un `opencode.json` de proyecto con los dos modelos fijados al local.
- Una forma de lanzar OpenCode sin arrastrar tu configuración global.
- Un script que te dice, en diez segundos, si hubo recortes, cuánto pesan los prompts y si algo salió a internet.

El resto del libro explica cada una de estas piezas y qué hacer cuando la tuya no se comporta igual.