

LOS 4 ENCHUFES DE CLAUDE CODE

Claude Code va lento si no le enchufas estas cuatro cosas. Qué hace cada una, el comando exacto para instalarla y el primer prompt para probarla.

01	**Supabase**	habla con tu base de datos	✓
02	**Vercel**	despliega con una frase	✓
03	**oh-my-claudecode**	19 agentes, un equipo	✓
04	**Chrome DevTools**	ojos en un navegador real	✓

Jose Hernández @jose.hernandez.ia

Qué vas a conseguir

Claude Code solo, escribe código. Con estos cuatro enchufes, además, toca tu base de datos, publica la app, reparte el trabajo entre agentes y comprueba el resultado en un navegador de verdad.

1. **Base de datos sin abrir el panel**
   Consultas, migraciones, tipos y revisión de seguridad desde la terminal.

2. **Desplegar con una frase**
   «Despliega mi app» y tienes una dirección de prueba o de producción.

3. **Un equipo en vez de un asistente**
   Escribes `/team` y varios agentes planifican, construyen, comprueban y corrigen.

4. **Ver lo que pasa de verdad**
   Errores de consola, peticiones de red, capturas y rendimiento de tu app.

> Todo lo de esta guía es gratis y oficial: el plugin de Supabase, el MCP de Vercel, el plugin de Google para Chrome y un proyecto abierto con licencia MIT.

Lo que necesitas

- [ ] **Claude Code** instalado y actualizado (`claude update`)
- [ ] **Node.js 22** o superior (`node -v` para comprobarlo)
- [ ] Una cuenta gratuita en **Supabase** y otra en **Vercel**, si vas a usar esos dos
- [ ] **Google Chrome** actualizado para el último enchufe
- [ ] Un proyecto de prueba, aunque sea una app de dos pantallas, para no practicar en producción

> Los dos servicios se conectan iniciando sesión en una ventana del navegador. No pegues claves ni contraseñas en el chat de Claude: si algo te las pide, para.

Enchufe 1: Supabase

El plugin oficial trae el servidor MCP de Supabase y sus skills de buenas prácticas: Claude ejecuta SQL, aplica migraciones, genera los tipos y te pasa los advisors de seguridad.

Instalar

```
npx plugins add supabase-community/supabase-plugin
```

Después, dentro de Claude Code, escribe `/mcp` y entra en tu cuenta de Supabase cuando se abra el navegador.

Primera prueba

```
Conéctate a mi proyecto de Supabase [nombre del proyecto]. Enséñame
las tablas con sus columnas y pásame los advisors de seguridad, cada
uno con el riesgo en una frase.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y
propón cómo mejorarlo.
```

Supabase sin sustos

1 **Empieza en solo lectura**
En el panel de Supabase, pestaña MCP, genera la URL de tu proyecto con la opción de solo lectura. Para leer y revisar no hace falta más.

2 **Un proyecto de desarrollo**
Haz las migraciones primero en una copia. Cuando funcionen, Claude te las deja en archivos para producción.

Una migración con red

```
Quiero añadir [una tabla de pedidos con cliente, importe y fecha].
Escríbeme la migración y, con mi visto bueno, aplícala en el
proyecto de desarrollo. Al terminar, regenera los tipos y vuelve a
pasar los advisors.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y
propón cómo mejorarlo.
```

> La base de datos de producción no se toca desde el chat hasta que la migración ha funcionado en la de desarrollo.

Enchufe 2: Vercel

El MCP oficial de Vercel despliega desde julio de 2026: crea el proyecto, detecta el framework, construye y te devuelve la dirección sin salir de la terminal.

Conectar

```
claude mcp add --transport http vercel https://mcp.vercel.com
```

Escribe `/mcp` en Claude Code y autoriza el acceso a tu cuenta de Vercel en el navegador. Solo la primera vez.

Primera prueba

```
Despliega esta app en Vercel como preview y pásame la dirección.
Antes, dime qué framework has detectado y qué comando de build vas a
usar. Si falla, enséñame el error antes de tocar nada.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y
propón cómo mejorarlo.
```

> Primero preview, siempre. A producción solo cuando hayas abierto la dirección de prueba y lo hayas visto con tus ojos.

Enchufe 3: oh-my-claudecode

Un proyecto abierto que convierte Claude Code en un equipo: 19 agentes especializados (ejecutores, depuradores, diseñadores...) que trabajan sobre una misma lista de tareas. Lo coordinas con un solo comando.

Instalar (dos comandos, dentro de Claude Code)

```
/plugin marketplace add https://github.com/Yeachan-Heo/oh-my-claudecode
/plugin install oh-my-claudecode
```

Lanzar un equipo

```
/team 3:executor "Arregla todos los errores de TypeScript del proyecto"
```

El número es cuántos agentes quieres. Si lo quitas, el propio sistema decide según el tamaño de la tarea. El equipo sigue siempre el mismo orden: planifica, construye, comprueba y corrige.

Un equipo que no se desmanda

Tarea para el equipo

```
Lanza un equipo de tres agentes para [pasar el formulario de
registro a validación en el servidor]. Antes de empezar, enséñame
el plan con qué hace cada uno. Al acabar, resumen de cambios,
tests que pasan y lo que queda sin resolver.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y
propón cómo mejorarlo.
```

1 **Empieza con dos o tres agentes**
Un equipo grande gasta mucho más y para la mayoría de tareas no hace falta.

2 **Pide el plan primero**
Es el momento barato de corregir el rumbo. Y `/oh-my-claudecode:cancel` lo para todo.

> Si el proyecto cambia la sintaxis del comando en una versión nueva, el README del repositorio manda sobre esta guía.

Enchufe 4: Chrome DevTools

Lo ha hecho el equipo de Chrome de Google. Le da a Claude un navegador real: errores de consola, peticiones de red, capturas y trazas de rendimiento. Se acabó programar a ciegas.

Instalar (dentro de Claude Code)

```
/plugin marketplace add ChromeDevTools/chrome-devtools-mcp
/plugin install chrome-devtools-mcp@chrome-devtools-plugins
```

Primera prueba

```
Abre [la dirección de tu preview] en el navegador. Dime los errores
de la consola, las peticiones que fallan o tardan más de un segundo
y graba una traza de rendimiento. Lista por gravedad y arregla
la primera.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y
propón cómo mejorarlo.
```

El flujo completo, enchufado

1. **Construye**
   La función nueva y su migración, enseñadas antes de aplicar.

2. **Preview en Vercel**
   Una dirección de prueba.

3. **DevTools la revisa y el equipo corrige**
   Consola, red y rendimiento; lo que falle lo arregla el equipo y DevTools vuelve a mirar.

4. **Producción**
   Solo con el segundo pase limpio.

Prompt maestro

```
Saca [la funcionalidad] a producción: construye y enséñame los cambios; preview en Vercel; revisión con Chrome DevTools; si falla algo, un equipo de tres agentes lo corrige y vuelves a revisar; cuando esté limpio, pídeme permiso para producción.

Pregúntame lo que dudes, enséñamelo antes de darlo por bueno y propón cómo mejorarlo.
```

Lo que nunca debes hacer

- [ ] Conectar Supabase en modo escritura a tu base de datos de producción el primer día
- [ ] Desplegar a producción sin haber abierto antes la preview
- [ ] Lanzar equipos de diez agentes para una tarea de media hora: paga quien no mira el coste
- [ ] Dejar que DevTools navegue con tu perfil personal de Chrome: usa uno aparte para Claude
- [ ] Pegar claves o tokens en el chat: las conexiones se hacen iniciando sesión en el navegador

FUENTES

Documentación de Supabase (plugins para Claude Code y servidor MCP), changelog de Vercel del 23 de julio de 2026 («Vercel MCP can now deploy code») y documentación de Vercel MCP, README de oh-my-claudecode (Yeachan-Heo, GitHub) y README de chrome-devtools-mcp (ChromeDevTools, GitHub). Comandos comprobados el 20 nov 2026.