cauce

Guía gratis · Claude Code

Las tres leyes: reglas que se cumplen siempre.

CLAUDE.md es un consejo. Un hook es una ley: lo corre Claude Code, no depende de que Claude se acuerde. Lo más fácil: pegale el bloque a Claude y pedile que lo agregue a tu archivo, así junta varios hooks sin romper nada.

01

Ley 1: se formatea solo.

`PostToolUse` `.claude/settings.json`

Después de cada edición, Prettier acomoda el archivo.

```
{
  "hooks": {
    "PostToolUse": [{
      "matcher": "Edit|Write",
      "hooks": [{
        "type": "command",
        "command": "jq -r
'.tool_input.file_path' | xargs npx prettier
--write"
      }]
    }]
  }
}
```

Necesita jq y Prettier instalados.

02

Ley 2: el guardia.

`.claude/hooks/protect-files.sh`

Este script frena la edición de archivos delicados. Salir con código 2 es lo que bloquea.

```
#!/bin/bash
INPUT=$(cat)
FILE_PATH=$(echo "$INPUT" | jq -r
'.tool_input.file_path // empty')
FILE_PATH="${FILE_PATH//\\//}"
PROTECTED_PATTERNS=(".env" "package-
lock.json" ".git/")
for pattern in "${PROTECTED_PATTERNS[@]}";
do
  if [[ "$FILE_PATH" == *"$pattern"* ]];
then
    echo "Blocked: $FILE_PATH matches
'$pattern'" >&2
    exit 2
  fi
done
exit 0
```

En macOS y Linux, hacelo ejecutable con chmod +x.

03

Ley 2: el registro.

`PreToolUse` `.claude/settings.json`

Corre el script antes de cada edición.

```
{
  "hooks": {
    "PreToolUse": [{
      "matcher": "Edit|Write",
      "hooks": [{
        "type": "command",
        "command":
"\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/prote
ct-files.sh"
      }]
    }]
  }
}
```

04

Ley 3: te avisa.

`Notification` `~/.claude/settings.json`

Cuando Claude te necesita, salta un aviso. Este es el de macOS.

```
{
  "hooks": {
    "Notification": [{
      "matcher": "",
      "hooks": [{
        "type": "command",
        "command": "osascript -e 'display
notification \"Claude te necesita\" with
title \"Claude Code\"'"
      }]
    }]
  }
}
```

05

Ley 3 en Windows y Linux.

`command`

Usá el mismo bloque y cambiá solo la línea command.

```
// Windows
"command": "powershell.exe -Command \"
[System.Reflection.Assembly]::LoadWithPartia
lName('System.Windows.Forms');
[System.Windows.Forms.MessageBox]::Show('Cla
ude te necesita', 'Claude Code')\""

// Linux
"command": "notify-send 'Claude Code'
'Claude te necesita'"
```

En Windows abre una ventanita, que puede quedar atrás de la terminal. La línea con // es solo para que ubiques cada una: no la pegues.

+

Leelo antes de pegar.

Un hook es un comando que corre en tu compu, con tus permisos. Nunca pegues uno que viste en un comentario sin entender qué hace.

Seguime para más trucos **@felipetizonmedia**

Fuente: code.claude.com/docs/en/hooks-guide (consultado 2026-11-09).