EcomLabs

GUÍA PASO A PASO

npm audit no es suficiente: controla tus dependencias

Los comandos para saber qué tienes instalado (npm ls), qué está desactualizado (npm outdated) y qué vas a agregar (npm view), cómo aprobar scripts de instalación en npm 12 y qué correr en CI. Con la checklist de dependencias.

PARA	Proyectos de Node.js que instalan con npm: tiendas, APIs, sitios en Next.js
PROBADO CON	Node 22.22.3 · npm 12.2.3, 11.21.3 y 11.11.1 · registro público de npm, 19 de noviembre de 2026
ACTUALIZADA	19 de noviembre de 2026
DOCUMENTACIÓN	docs.npmjs.com/cli/v12
ARTÍCULO	ecomlabs.dev/g/dependencias
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué versión de npm tienes
Paso 01	Qué tienes instalado: npm ls
Paso 02	Qué está desactualizado: npm outdated
Paso 03	Qué vas a agregar: npm view
Paso 04	Scripts de instalación
Paso 05	Qué correr en CI
Al terminar	Checklist de dependencias

ANTES DE EMPEZAR

Qué versión de npm tienes

Corre `npm -v`. Desde npm 12 (8 de julio de 2026), los scripts de instalación de las dependencias no corren hasta que los apruebas en `allowScripts`. En npm 11 corren: la 11.21.3 avisa, la 11.11.1 ni eso. npm 12 pide Node `^22.22.2 || ^24.15.0 || >=26.0.0` y deja de instalar dependencias desde git o desde URLs de tarball hasta que lo permitas (`allow-git`, `allow-remote`).

npm audit solo cruza los nombres y versiones de tu árbol con avisos ya publicados. Lo que no tiene aviso no aparece; los pasos de esta guía cubren el resto.

PASO 01

Qué tienes instalado: npm ls

```
npm ls                          # solo tus dependencias directas
npm ls --all                    # el árbol completo
npm ls qs                       # dónde aparece un paquete y en qué versión
npm explain qs                  # qué dependencia directa lo trajo
npm ls --all --parseable | wc -l   # cuántos paquetes hay (resta 1: tu proyecto)
```

- En la salida, `deduped` significa que el paquete se comparte con otra rama y se instaló una sola vez.
- Si un aviso menciona un paquete que no reconoces, `npm explain` te dice qué dependencia directa actualizar.
- Para medir un paquete antes de agregarlo: carpeta vacía, `npm init -y`, `npm install <paquete> --package-lock-only` y `npm ls --all --parseable --package-lock-only`. Así medimos 65 paquetes para `express@5.3.0` y 50 para `express@4.17.1`.

PASO 02

Qué está desactualizado: npm outdated

```
$ npm outdated
Package  Current  Wanted  Latest  Location             Depended by
express   4.17.1  4.22.3   5.3.0  node_modules/express  demo-store
```

- `Current` es lo instalado. `Wanted` es lo más nuevo que acepta tu rango: súbelo con `npm update express`. `Latest` es la última publicada: si es versión mayor, lee las notas y súbela sola con `npm install express@5`.
- Sale con código 1 si hay algo desactualizado: en CI úsalo como reporte, no como bloqueo.
- En la prueba, `npm update express` agregó 18 paquetes y cambió 23. Revisa el diff del lockfile.
- Una dependencia por commit, con el lockfile en el mismo commit, y pruebas entre cada una.
- `min-release-age=7` en el `.npmrc` evita adoptar versiones con menos de 7 días. Con 60, npm eligió express 5.2.1 en vez de 5.3.0. Los parches de seguridad urgentes se instalan sin esperar.

PASO 03

Qué vas a agregar: npm view

`npm view` lee el registro público sin instalar nada. Con varios campos imprime `campo = valor`, por ejemplo `npm view express dist-tags.latest time.modified repository.url`.

COMANDO	QUÉ REVISAS	SEÑAL DE ALERTA
`npm view <pkg> maintainers`	Quién puede publicar	Una sola cuenta
`npm view <pkg> time.modified`	Último cambio	Años sin cambios y avisos abiertos
`npm view <pkg> repository.url`	Dónde está el código	Sin repositorio
`npm view <pkg> deprecated`	Si está deprecado	Devuelve un mensaje
`npm view <pkg> scripts`	Scripts al instalar	`preinstall`, `install`, `postinstall`
`npm view <pkg> dependencies`	Dependencias directas	Decenas para algo pequeño

En npm 12, `npm view --json` siempre devuelve un arreglo (`["5.3.0"]`); npm 11 devolvía el valor suelto. Copia el nombre del paquete desde su documentación oficial: un nombre casi igual al que buscabas es un patrón conocido de paquetes maliciosos.

PASO 04

Scripts de instalación

```
npm install-scripts ls                # lo pendiente de revisar
npm view esbuild@0.25.0 scripts       # lee el script antes de decidir
npm install-scripts approve esbuild   # escribe "esbuild@0.25.0": true
npm rebuild                           # corre los scripts recién aprobados
npm install-scripts deny esbuild      # escribe "esbuild": false (todas las versiones)
```

- La aprobación queda fijada a la versión: al subir de versión, el script vuelve a bloquearse hasta aprobarlo de nuevo.
- `approve` y `deny` trabajan sobre lo instalado; si el paquete no está, fallan con `ENOMATCH`.
- Muchos paquetes funcionan sin su script; un módulo nativo que compila al instalar no.
- No uses `approve --all` ni `--dangerously-allow-all-scripts` para quitar el aviso.
- Con la lista revisada, pon `strict-allow-scripts=true` en el `.npmrc`: lo no revisado falla con `ESTRICTALLOWSCRIPTS`.
- En npm 11 los scripts corren. La 11.21.3 ya trae `npm install-scripts` para preparar la lista; el script `list-install-scripts.mjs` del artículo los lista en cualquier versión.

PASO 05

Qué correr en CI

```
npm ci
npm audit --audit-level=high
npm audit signatures
```

- `npm ci` exige el lockfile, falla si no coincide con `package.json` y nunca escribe en esos archivos.
- `--audit-level=high` solo cambia cuándo falla el paso; no filtra el reporte, así que léelo completo.
- `npm audit signatures` verifica firmas del registro y atestaciones de procedencia.
- No pongas `npm audit fix --force` en CI: puede instalar versiones mayores.

AL TERMINAR

Checklist de dependencias

- [ ] `package-lock.json` está commiteado y CI instala con `npm ci`.
- [ ] `npm audit --audit-level=high` y `npm audit signatures` corren en CI.
- [ ] Sabes cuántos paquetes tiene tu árbol (`npm ls --all --parseable`).
- [ ] Antes de agregar un paquete lo revisas con `npm view` y copias su nombre de la documentación oficial.
- [ ] Revisas `npm outdated` con una frecuencia fija y actualizas una dependencia por commit.
- [ ] Las versiones mayores van solas, con las notas leídas y las pruebas corriendo.
- [ ] `min-release-age` está en el `.npmrc` y los parches de seguridad se aplican sin esperar.
- [ ] En npm 12, `allowScripts` solo tiene paquetes revisados y `strict-allow-scripts=true` está activo.