EcomLabs

STEP-BY-STEP GUIDE

# npm audit is not enough: control your dependencies

The commands to see what you have installed (npm ls), what is out of date (npm outdated) and what you are about to add (npm view), how to approve install scripts in npm 12 and what to run in CI. With the dependency checklist.

| | |
|---|---|
| FOR | Node.js projects that install with npm: stores, APIs, Next.js sites |
| TESTED WITH | Node 22.22.3 · npm 12.2.3, 11.21.3 and 11.11.1 · public npm registry, November 19, 2026 |
| UPDATED | November 19, 2026 |
| DOCS | docs.npmjs.com/cli/v12 |
| ARTICLE | ecomlabs.dev/g/dependencias-en |
| AUTHOR | Luis Morales · @luismoralesdev |

# Contents


| | |
|---|---|
| Before you start | Which npm version you have |
| Step 01 | What you have installed: npm ls |
| Step 02 | What is out of date: npm outdated |
| Step 03 | What you are about to add: npm view |
| Step 04 | Install scripts |
| Step 05 | What to run in CI |
| When you are done | Dependency checklist |

BEFORE YOU START

## Which npm version you have

Run `npm -v`. Since npm 12 (July 8, 2026), dependency install scripts do not run until you approve them in `allowScripts`. On npm 11 they run: 11.21.3 warns, 11.11.1 does not even do that. npm 12 requires Node `^22.22.2 || ^24.15.0 || >=26.0.0` and stops installing dependencies from git or tarball URLs until you allow it (`allow-git`, `allow-remote`).

npm audit only matches the names and versions in your tree against published advisories. Anything without an advisory does not show up; the steps in this guide cover the rest.

STEP 01

## What you have installed: npm ls

```
npm ls                          # only your direct dependencies
npm ls --all                    # the whole tree
npm ls qs                       # where a package appears and which version
npm explain qs                  # which direct dependency brought it in
npm ls --all --parseable | wc -l   # how many packages (subtract 1: your project)
```

- In the output, `deduped` means the package is shared with another branch and was installed once.
- If an advisory names a package you do not recognize, `npm explain` tells you which direct dependency to update.
- To size up a package before adding it: empty folder, `npm init -y`, `npm install <package> --package-lock-only` and `npm ls --all --parseable --package-lock-only`. That is how we measured 65 packages for `express@5.3.0` and 50 for `express@4.17.1`.

STEP 02

## What is out of date: npm outdated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$ npm outdated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` is what is installed. `Wanted` is the newest your range accepts: move to it with `npm update express`. `Latest` is the newest published: if it is a major, read the notes and upgrade it alone with `npm install express@5`.
- It exits with code 1 when anything is behind: in CI, use it as a report, not a gate.
- In the test, `npm update express` added 18 packages and changed 23. Review the lockfile diff.
- One dependency per commit, with the lockfile in the same commit, and tests between each one.
- `min-release-age=7` in `.npmrc` avoids adopting versions less than 7 days old. With 60, npm picked express 5.2.1 instead of 5.3.0. Urgent security patches go in without waiting.

STEP 03

## What you are about to add: npm view

`npm view` reads the public registry without installing anything. With several fields it prints `field = value`, for example `npm view express dist-tags.latest time.modified repository.url`.

| COMMAND | WHAT YOU CHECK | RED FLAG |
|---|---|---|
| `npm view <pkg> maintainers` | Who can publish | A single account |
| `npm view <pkg> time.modified` | Last change | Years idle plus open advisories |
| `npm view <pkg> repository.url` | Where the code lives | No repository |
| `npm view <pkg> deprecated` | Whether it is deprecated | It returns a message |
| `npm view <pkg> scripts` | Scripts on install | `preinstall`, `install`, `postinstall` |
| `npm view <pkg> dependencies` | Direct dependencies | Dozens for a small job |

In npm 12, `npm view --json` always returns an array (`["5.3.0"]`); npm 11 returned the bare value. Copy the package name from its official documentation: a name almost identical to the one you wanted is a known pattern for malicious packages.

STEP 04

## Install scripts

```
npm install-scripts ls                # what is pending review
npm view esbuild@0.25.0 scripts       # read the script before deciding
npm install-scripts approve esbuild   # writes "esbuild@0.25.0": true
npm rebuild                           # runs the newly approved scripts
npm install-scripts deny esbuild      # writes "esbuild": false (every version)
```

- The approval is pinned to the version: after an upgrade, the script is blocked again until approved again.
- `approve` and `deny` work on what is installed; if the package is not there, they fail with `ENOMATCH`.
- Many packages work without their script; a native module that compiles on install does not.
- Do not use `approve --all` or `--dangerously-allow-all-scripts` to silence the warning.
- Once the list is reviewed, set `strict-allow-scripts=true` in `.npmrc`: anything unreviewed fails with `ESTRICTALLOWSCRIPTS`.
- On npm 11 the scripts run. 11.21.3 already ships `npm install-scripts` to prepare the list; the article's `list-install-scripts.mjs` lists them on any version.

STEP 05

## What to run in CI

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

- `npm ci` requires the lockfile, fails when it does not match `package.json` and never writes to those files.
- `--audit-level=high` only changes when the step fails; it does not filter the report, so read all of it.
- `npm audit signatures` verifies registry signatures and provenance attestations.
- Do not put `npm audit fix --force` in CI: it can install major versions.

WHEN YOU ARE DONE

## Dependency checklist

- [ ] `package-lock.json` is committed and CI installs with `npm ci`.
- [ ] `npm audit --audit-level=high` and `npm audit signatures` run in CI.
- [ ] You know how many packages your tree has (`npm ls --all --parseable`).
- [ ] Before adding a package you check it with `npm view` and copy its name from the official docs.
- [ ] You check `npm outdated` on a fixed schedule and update one dependency per commit.
- [ ] Major versions go alone, with the notes read and the tests running.
- [ ] `min-release-age` is in `.npmrc` and security patches go in without waiting.
- [ ] On npm 12, `allowScripts` only lists reviewed packages and `strict-allow-scripts=true` is on.