GUÍA PASO A PASO

Actualizar Node en Docker y en tu plataforma

Cómo saber qué Node corre de verdad en producción, cómo cambiarlo en un Dockerfile, en Railway (Nixpacks, Railpack o Dockerfile), Render, Fly.io, Vercel y Heroku, cómo reconstruir sin arrastrar la imagen vieja y qué hacer si sigues en Node 20.

PARA	Apps Node.js y Next.js en Docker, Railway y otras plataformas
PROBADO CON	Node 22.22.3 y 24.22.2 · Docker CLI 28.3.2 (sin motor) · Windows 11
ACTUALIZADA	3 de noviembre de 2026
DOCUMENTACIÓN	nodejs.org/en/blog/vulnerability/october-2026-security-releases
ARTÍCULO	ecomlabs.dev/g/node
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué cubre esta guía
Paso 01	Averigua qué Node corre de verdad
Paso 02	Compara con la versión con parche
Paso 03	Docker: cambia la imagen y reconstruye
Paso 04	Railway: Nixpacks, Railpack o Dockerfile
Paso 05	Otras plataformas
Paso 06	Si sigues en Node 20
Al terminar	Checklist

ANTES DE EMPEZAR

Qué cubre esta guía

El 21 de octubre de 2026, Node.js publicó parches de seguridad para las líneas 26.x, 24.x y 22.x: diez vulnerabilidades, cuatro de severidad alta. Actualizar Next.js o cualquier paquete de npm no cambia Node: el runtime viene de la imagen base de Docker o de lo que instala tu plataforma. Esta guía es el procedimiento para cambiarlo y comprobarlo. Lo de cada plataforma viene de su documentación, leída el 2 de noviembre.

LÍNEA	MÍNIMA CON PARCHE	ÚLTIMA AL 2 NOV	FIN DE SOPORTE
26.x	26.12.1	26.13.0	30 abr 2029
24.x	24.22.1	24.22.2	30 abr 2028
22.x	22.23.4	22.23.4	30 abr 2027
20.x	Ninguna (End-of-Life)	20.20.2	30 abr 2026

PASO 01

Averigua qué Node corre de verdad

El `node -v` de tu computadora no cuenta. Lo que cuenta es el Node de la imagen o del servidor:

```
docker run --rm mi-app node -v
docker buildx imagetools inspect node:24-bookworm-slim --format '{{json (index .Image
"linux/amd64").Config.Env}}'
```

La primera línea imprime el Node de tu imagen. La segunda lee del registro qué Node trae una etiqueta, sin bajarla y sin motor de Docker: el 2 de noviembre devolvió `NODE_VERSION=24.22.2`. En producción, haz que la app imprima `process.version` al arrancar; en nuestra prueba con Node puro salió `[startup] Node v22.22.3 · win32-x64 · OpenSSL 3.5.6`. En Next.js va en `instrumentation.ts`, en la raíz o en `src` (según su documentación, `register()` se llama una vez al arrancar cada servidor; esta variante no la ejecutamos):

```
export function register() {
  if (process.env.NEXT_RUNTIME === "nodejs") {
    console.log(`[startup] Node ${process.version}`);
  }
}
```

No publiques la versión en un endpoint abierto: le dice a cualquiera si estás sin parche. Y no te fíes del log de build: con Nixpacks en Railway dice `setup | nodejs_24`, sin el parche.

PASO 02

Compara con la versión con parche

`check-node.mjs` compara una versión con la mínima corregida de su línea. Sin argumentos revisa el Node que lo ejecuta; con uno (`node check-node.mjs v24.10.0`), la versión que le pases. Sale con código 1 si falta el parche, así que sirve en CI.

```
// Lowest versions with the fixes from the Node.js security release of October 21, 2026.
const PATCHED = { 22: "22.23.4", 24: "24.22.1", 26: "26.12.1" };

const version = (process.argv[2] ?? process.version).trim().replace(/^v/, "");
const [major, minor, patch] = version.split(".").map((n) => parseInt(n, 10));
const min = PATCHED[major];

if (!min) {
  console.error(`Node ${version}: ${major}.x is not one of the lines patched on October 21, 2026 (22, 24, 26).`);
  process.exit(1);
}

const [, minMinor, minPatch] = min.split(".").map(Number);
if (minor < minMinor || (minor === minMinor && patch < minPatch)) {
  console.error(`Node ${version} is older than ${min}: update the runtime, not just your dependencies.`);
  process.exit(1);
}

console.log(`Node ${version} has the October 21, 2026 fixes (${min} or later).`);
```

Probado con Node 22.22.3 y 24.22.2 en Windows 11: 22.22.3, 24.10.0 y 20.20.2 salen con 1; 22.23.4, 24.22.1 y 26.13.0, con 0. En el próximo release de seguridad, actualiza `PATCHED`.

PASO 03

Docker: cambia la imagen y reconstruye

Pon la versión en un `ARG` antes del primer `FROM`. Es el patrón estándar de dos etapas para Next.js; no lo construimos, porque en nuestro equipo el motor de Docker no arranca:

```
ARG NODE_VERSION=24.22.2

FROM node:${NODE_VERSION}-bookworm-slim AS build
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci
COPY . .
RUN npm run build

FROM node:${NODE_VERSION}-bookworm-slim
WORKDIR /app
```

```
ENV NODE_ENV=production
COPY --from=build --chown=node:node /app ./
USER node
CMD ["node", "node_modules/next/dist/bin/next", "start"]
```

- Agrega un `.dockerignore` con `node_modules` y `.next`, para que `COPY . .` no pise lo que instaló `npm ci`.
- Mantén tu variante: `-slim` (lo mínimo para correr `node`), `-alpine` (más pequeña, con musl en lugar de glibc) o la predeterminada. `bookworm` y `trixie` son versiones de Debian.
- Etiqueta exacta (`node:24.22.2-bookworm-slim`): siempre esa versión de Node. Flotante (`node:24-bookworm-slim`): sube con la línea 24 solo si reconstruyes con `--pull`. `node:20-bookworm-slim` no cambia desde el 22 de abril de 2026.
- Para fijar la imagen exacta, agrega el digest (el 2 de noviembre era `sha256:c86ecead9e55a5210ac1727de38a4ca840bf03dcaf34dc9749addbf383b1920c`). Docker advierte que así renuncias a las correcciones de seguridad automáticas.

```
docker build --pull -t mi-app .
docker build --pull --no-cache -t mi-app .
docker run --rm mi-app node -v
```

`--pull` busca una imagen base más nueva aunque haya una en caché; `--no-cache` rehace las capas, pero no baja la imagen base. Despliega el build nuevo: redesplegar el anterior trae el Node anterior.

PASO 04

Railway: Nixpacks, Railpack o Dockerfile

Con Nixpacks (`"builder": "NIXPACKS"` en `railway.json`) solo eliges la versión mayor, con `NIXPACKS_NODE_VERSION`, `engines.node` o `.nvmrc`. El parche lo fija un commit de nixpkgs dentro de Nixpacks: según su código, para 24 es `23f9169c…`, y en ese commit nixpkgs define Node 24.10.0. Reconstruir no lo cambia. Nixpacks está en modo de mantenimiento y su README recomienda Railpack. Dos salidas:

1. **Railpack**, el builder por defecto de Railway desde el 19 de septiembre de 2025: `"builder": "RAILPACK"` en `railway.json`. Lee la versión, en este orden, de `RAILPACK_NODE_VERSION`, `devEngines.runtime`, `engines.node`, `.nvmrc`, `.node-version`, `mise.toml` o `.tool-versions`; si no, usa `lts`. Su documentación solo muestra versiones mayores (`22`): confirma el parche en el log de arranque.
2. **Dockerfile**: Railway usa un `Dockerfile` en la raíz si lo encuentra (el log dice `Using detected Dockerfile!`), u otro con `RAILWAY_DOCKERFILE_PATH`. La versión es la del `FROM`.

> **CUIDADO**
>
> Cambiar de builder cambia todo el build, no solo Node. Pruébalo primero en un entorno aparte de Railway, revisa el log de build y el de arranque, y solo entonces pásalo a producción.

PASO 05

Otras plataformas

PLATAFORMA	DÓNDE SE ELIGE	QUÉ HACER
Render	`NODE_VERSION`, `.node-version`, `.nvmrc`, `engines`	Acepta versión exacta o alias (`lts`); 24.22.1 por defecto en servicios creados desde el 21 oct. Pon límite superior a los rangos.
Fly.io	`ARG NODE_VERSION` del Dockerfile generado	El generador copia el Node de tu computadora. Cambia el `ARG` y despliega.
Vercel	Settings o `engines.node` (solo mayor)	Vercel aplica los parches. Comprueba con `node -v` en el build.
Heroku	`engines.node`	Recomienda `24.x`; evita `>= 24.x`. Despliega un build nuevo.

PASO 06

Si sigues en Node 20

Node 20 está en End-of-Life desde el 30 de abril de 2026; su última versión es 20.20.2. El aviso de octubre dice que las versiones End-of-Life siempre están afectadas cuando hay un release de seguridad.

1. Elige destino: 22.x (soporte hasta el 30 abr 2027), 24.x (hasta el 30 abr 2028) o 26.x (LTS desde el 28 oct, hasta el 30 abr 2029). Node.js pide usar en producción solo versiones Active LTS o Maintenance LTS.
2. En una rama: instalación limpia con la versión nueva, build, pruebas y los flujos críticos de tu tienda. Luego, pasos 03 a 05.
3. Si no puedes migrar ya, Node.js menciona soporte comercial de HeroDevs, NodeSource y TuxCare como solución temporal.

AL TERMINAR

Checklist

- [ ] Sabes qué Node corre en producción, por el log de arranque o por `node -v` en el contenedor.
- [ ] Es 22.23.4, 24.22.1, 26.12.1 o posterior: `check-node.mjs` sale con código 0.
- [ ] La versión está escrita en un solo lugar: el `ARG` del Dockerfile o la configuración de la plataforma.
- [ ] Reconstruiste con `--pull` y desplegaste el build nuevo, no el anterior.
- [ ] Si usas Nixpacks en Railway, probaste Railpack o un Dockerfile en un entorno aparte.
- [ ] Si sigues en Node 20, tienes fecha para pasar a 22, 24 o 26.
- [ ] Tienes un recordatorio para el próximo release de seguridad de Node.js.