EcomLabs

STEP-BY-STEP GUIDE

Update Node in Docker and on your platform

How to find the Node version that really runs in production, how to change it in a Dockerfile, on Railway (Nixpacks, Railpack or a Dockerfile), Render, Fly.io, Vercel and Heroku, how to rebuild without dragging the old image along, and what to do if you're still on Node 20.

FOR	Node.js and Next.js apps on Docker, Railway and other platforms
TESTED WITH	Node 22.22.3 and 24.22.2 · Docker CLI 28.3.2 (no engine) · Windows 11
UPDATED	November 3, 2026
DOCS	nodejs.org/en/blog/vulnerability/october-2026-security-releases
ARTICLE	ecomlabs.dev/g/node-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What this guide covers
Step 01	Find the Node version that really runs
Step 02	Compare it with the patched version
Step 03	Docker: change the image and rebuild
Step 04	Railway: Nixpacks, Railpack or a Dockerfile
Step 05	Other platforms
Step 06	If you're still on Node 20
When you're done	Checklist

BEFORE YOU START

What this guide covers

On October 21, 2026, Node.js shipped security fixes for the 26.x, 24.x and 22.x lines: ten vulnerabilities, four of them high severity. Upgrading Next.js or any npm package doesn't change Node: the runtime comes from your Docker base image or from whatever your platform installs. This guide is the procedure to change it and confirm it. Platform details come from each platform's docs, read on November 2.

LINE	MINIMUM PATCHED	LATEST ON NOV 2	END OF LIFE
26.x	26.12.1	26.13.0	Apr 30, 2029
24.x	24.22.1	24.22.2	Apr 30, 2028
22.x	22.23.4	22.23.4	Apr 30, 2027
20.x	None (end-of-life)	20.20.2	Apr 30, 2026

STEP 01

Find the Node version that really runs

Your laptop's `node -v` doesn't count. What counts is the Node in the image or on the server:

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

The first line prints your image's Node. The second reads which Node a tag ships straight from the registry, without pulling it and without a Docker engine: on November 2 it returned `NODE_VERSION=24.22.2`. In production, have the app print `process.version` at startup; our plain-Node test printed `[startup] Node v22.22.3 · win32-x64 · OpenSSL 3.5.6`. In Next.js it goes in `instrumentation.ts`, at the root or in `src` (per its docs, `register()` is called once when each server starts; we did not run this variant):

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

Don't publish the version on an open endpoint: it tells anyone whether you're unpatched. And don't trust the build log: with Nixpacks on Railway it says `setup | nodejs_24`, with no patch version.

STEP 02

Compare it with the patched version

`check-node.mjs` compares a version with the minimum patched release of its line. With no arguments it checks the Node running it; with one (`node check-node.mjs v24.10.0`), the version you pass. It exits with code 1 when the fix is missing, so it works in 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).`);
```

Tested with Node 22.22.3 and 24.22.2 on Windows 11: 22.22.3, 24.10.0 and 20.20.2 exit with 1; 22.23.4, 24.22.1 and 26.13.0 with 0. At the next security release, update `PATCHED`.

STEP 03

Docker: change the image and rebuild

Put the version in an `ARG` before the first `FROM`. This is the standard two-stage pattern for Next.js; we did not build it, because the Docker engine doesn't start on our machine:

```
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"]
```

- Add a `.dockerignore` with `node_modules` and `.next`, so `COPY . .` doesn't overwrite what `npm ci` installed.
- Keep your variant: `-slim` (the minimum to run `node`), `-alpine` (smaller, with musl instead of glibc) or the default. `bookworm` and `trixie` are Debian releases.
- Exact tag (`node:24.22.2-bookworm-slim`): always that Node version. Floating tag (`node:24-bookworm-slim`): moves with the 24 line only if you rebuild with `--pull`. `node:20-bookworm-slim` hasn't changed since April 22, 2026.
- To pin the exact image, add the digest (on November 2 it was `sha256:c86ecead9e55a5210ac1727de38a4ca840bf03dcaf34dc9749addbf383b1920c`). Docker warns that you then opt out of automated security fixes.

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

`--pull` checks for a newer base image even when one is cached; `--no-cache` rebuilds the layers but doesn't pull the base image. Deploy the new build: redeploying the old one brings back the old Node.

STEP 04

Railway: Nixpacks, Railpack or a Dockerfile

With Nixpacks (`"builder": "NIXPACKS"` in `railway.json`) you only choose the major version, through `NIXPACKS_NODE_VERSION`, `engines.node` or `.nvmrc`. The patch is pinned by a nixpkgs commit inside Nixpacks: per its source, for 24 it's `23f9169c…`, and at that commit nixpkgs defines Node 24.10.0. Rebuilding doesn't change it. Nixpacks is in maintenance mode, and its README recommends Railpack. Two ways out:

1. **Railpack**, Railway's default builder since September 19, 2025: `"builder": "RAILPACK"` in `railway.json`. It reads the version, in this order, from `RAILPACK_NODE_VERSION`, `devEngines.runtime`, `engines.node`, `.nvmrc`, `.node-version`, `mise.toml` or `.tool-versions`; otherwise it uses `lts`. Its docs only show major versions (`22`): confirm the patch in the startup log.
2. **A Dockerfile**: Railway uses a `Dockerfile` at the root when it finds one (the log says `Using detected Dockerfile!`), or another one through `RAILWAY_DOCKERFILE_PATH`. The version is the one in `FROM`.

> **CAUTION**
>
> Switching builders changes the whole build, not just Node. Try it first in a separate Railway environment, check the build log and the startup log, and only then move it to production.

STEP 05

Other platforms

PLATFORM	WHERE YOU SET IT	WHAT TO DO
Render	`NODE_VERSION`, `.node-version`, `.nvmrc`, `engines`	Takes an exact version or an alias (`lts`); 24.22.1 by default for services created since Oct 21. Put an upper bound on ranges.
Fly.io	`ARG NODE_VERSION` in the generated Dockerfile	The generator copies your machine's Node. Change the `ARG` and deploy.
Vercel	Settings or `engines.node` (major only)	Vercel rolls out patches. Check with `node -v` in the build.
Heroku	`engines.node`	Recommends `24.x`; avoid `>= 24.x`. Deploy a new build.

STEP 06

If you're still on Node 20

Node 20 has been end-of-life since April 30, 2026; its last release is 20.20.2. The October advisory says end-of-life versions are always affected when a security release occurs.

1. Pick a target: 22.x (supported until Apr 30, 2027), 24.x (until Apr 30, 2028) or 26.x (LTS since Oct 28, until Apr 30, 2029). Node.js says production apps should only use Active LTS or Maintenance LTS releases.
2. On a branch: a clean install with the new version, the build, your tests and your store's critical flows. Then steps 03 to 05.
3. If you can't migrate yet, Node.js lists commercial support from HeroDevs, NodeSource and TuxCare as a temporary solution.

WHEN YOU'RE DONE

Checklist

- [ ] You know which Node runs in production, from the startup log or `node -v` in the container.
- [ ] It's 22.23.4, 24.22.1, 26.12.1 or later: `check-node.mjs` exits with code 0.
- [ ] The version is written in one place: the Dockerfile `ARG` or the platform setting.
- [ ] You rebuilt with `--pull` and deployed the new build, not the previous one.
- [ ] If you use Nixpacks on Railway, you tried Railpack or a Dockerfile in a separate environment.
- [ ] If you're on Node 20, you have a date to move to 22, 24 or 26.
- [ ] You have a reminder for the next Node.js security release.