Migration guide

This article provides a set of guidelines for migrating from Nest version 9 to version 10. To learn more about the new features we've added in v10, check out this article. There were some very minor breaking changes that shouldn't affect most users - you can find the full list of them here.

Upgrading packages

While you can upgrade your packages manually, we recommend using ncu (npm check updates).

Cache module

The `CacheModule` has been removed from the `@nestjs/common` package and is now available as a standalone package - `@nestjs/cache-manager`. This change was made to avoid unnecessary dependencies in the `@nestjs/common` package. You can learn more about the `@nestjs/cache-manager` package here.

Deprecations

All deprecated methods & modules have been removed.

CLI Plugins and TypeScript >= 4.8

NestJS CLI Plugins (available for `@nestjs/swagger` and `@nestjs/graphql` packages) will now require TypeScript >= v4.8 and so older versions of TypeScript will no longer be supported. The reason for this change is that in TypeScript v4.8 introduced several breaking changes in its Abstract Syntax Tree (AST) which we use to auto-generate OpenAPI and GraphQL schemas.

Dropping support for Node.js v12

As of NestJS 10, we no longer support Node.js v12, as v12 went EOL on April 30, 2022. This means that NestJS 10 requires Node.js v16 or higher. This decision was made to allow us to finally set target to `ES2021` in our TypeScript configuration, instead of shipping polyfills as we did in the past.

From now on, every official NestJS package will be compiled to `ES2021` by default, which should result in a smaller library size and sometimes even (slightly) better performance.

We also strongly recommend using the latest LTS version.