CLI command reference

nest new

Creates a new (standard mode) Nest project.

```
$ nest new <name> [options]
$ nest n <name> [options]
```

Description

Creates and initializes a new Nest project. Prompts for package manager.

- Creates a folder with the given `<name>`
- Populates the folder with configuration files
- Creates sub-folders for source code (`/src`) and end-to-end tests (`/test`)
- Populates the sub-folders with default files for app components and tests

Arguments

Argument	Description
`<name>`	The name of the new project

Options

Option	Description
`--dry-run`	Reports changes that would be made, but does not change the filesystem. Alias: `-d`
`--skip-git`	Skip git repository initialization. Alias: `-g`
`--skip-install`	Skip package installation. Alias: `-s`
`--package-manager [package-manager]`	Specify package manager. Use `npm`, `yarn`, or `pnpm`. Package manager must be installed globally. Alias: `-p`
`--language [language]`	Specify programming language (`TS` or `JS`). Alias: `-l`
`--collection [collectionName]`	Specify schematics collection. Use package name of installed npm package containing schematic. Alias: `-c`

Option	Description
`--strict`	Start the project with the following TypeScript compiler flags enabled: `strictNullChecks`, `noImplicitAny`, `strictBindCallApply`, `forceConsistentCasingInFileNames`, `noFallthroughCasesInSwitch`

nest generate

Generates and/or modifies files based on a schematic

```
$ nest generate <schematic> <name> [options]
$ nest g <schematic> <name> [options]
```

Arguments

Argument	Description
`<schematic>`	The `schematic` or `collection:schematic` to generate. See the table below for the available schematics.
`<name>`	The name of the generated component.

Schematics

Name	Alias	Description
`app`		Generate a new application within a monorepo (converting to monorepo if it's a standard structure).
`library`	`lib`	Generate a new library within a monorepo (converting to monorepo if it's a standard structure).
`class`	`cl`	Generate a new class.
`controller`	`co`	Generate a controller declaration.
`decorator`	`d`	Generate a custom decorator.
`filter`	`f`	Generate a filter declaration.
`gateway`	`ga`	Generate a gateway declaration.
`guard`	`gu`	Generate a guard declaration.
`interface`	`itf`	Generate an interface.
`interceptor`	`itc`	Generate an interceptor declaration.
`middleware`	`mi`	Generate a middleware declaration.
`module`	`mo`	Generate a module declaration.

Name	Alias	Description
`pipe`	`pi`	Generate a pipe declaration.
`provider`	`pr`	Generate a provider declaration.
`resolver`	`r`	Generate a resolver declaration.
`resource`	`res`	Generate a new CRUD resource. See the CRUD (resource) generator for more details.
`service`	`s`	Generate a service declaration.

Options

Option	Description
`--dry-run`	Reports changes that would be made, but does not change the filesystem. Alias: `-d`
`--project [project]`	Project that element should be added to. Alias: `-p`
`--flat`	Do not generate a folder for the element.
`--collection [collectionName]`	Specify schematics collection. Use package name of installed npm package containing schematic. Alias: `-c`
`--spec`	Enforce spec files generation (default)
`--no-spec`	Disable spec files generation

nest build

Compiles an application or workspace into an output folder.

Also, the `build` command is responsible for:

- mapping paths (if using path aliases) via `tsconfig-paths`
- annotating DTOs with OpenAPI decorators (if `@nestjs/swagger` CLI plugin is enabled)
- annotating DTOs with GraphQL decorators (if `@nestjs/graphql` CLI plugin is enabled)

```
$ nest build <name> [options]
```

Arguments

Argument	Description
`<name>`	The name of the project to build.

Options

Option	Description
`--path [path]`	Path to `tsconfig` file. Alias `-p`
`--config [path]`	Path to `nest-cli` configuration file. Alias `-c`
`--watch`	Run in watch mode (live-reload). If you're using `tsc` for compilation, you can type `rs` to restart the application (when `manualRestart` option is set to `true`). Alias `-w`
`--builder [name]`	Specify the builder to use for compilation (`tsc`, `swc`, or `webpack`). Alias `-b`
`--webpack`	Use webpack for compilation (deprecated: use `--builder webpack` instead).
`--webpackPath`	Path to webpack configuration.
`--tsc`	Force use `tsc` for compilation.

nest start

Compiles and runs an application (or default project in a workspace).

```
$ nest start <name> [options]
```

Arguments

Argument	Description
`<name>`	The name of the project to run.

Options

Option	Description
`--path [path]`	Path to `tsconfig` file. Alias `-p`
`--config [path]`	Path to `nest-cli` configuration file. Alias `-c`
`--watch`	Run in watch mode (live-reload) Alias `-w`

Option	Description
`--builder [name]`	Specify the builder to use for compilation (`tsc`, `swc`, or `webpack`). Alias `-b`
`--preserveWatchOutput`	Keep outdated console output in watch mode instead of clearing the screen. (`tsc` watch mode only)
`--watchAssets`	Run in watch mode (live-reload), watching non-TS files (assets). See Assets for more details.
`--debug [hostport]`	Run in debug mode (with --inspect flag) Alias `-d`
`--webpack`	Use webpack for compilation. (deprecated: use `--builder webpack` instead)
`--webpackPath`	Path to webpack configuration.
`--tsc`	Force use `tsc` for compilation.
`--exec [binary]`	Binary to run (default: `node`). Alias `-e`

nest add

Imports a library that has been packaged as a **nest library**, running its install schematic.

```
$ nest add <name> [options]
```

Arguments

Argument	Description
`<name>`	The name of the library to import.

nest info

Displays information about installed nest packages and other helpful system info. For example:

```
$ nest info
```

```
 _   _           _      ___  _____  _____  _     _____
| \ | |         | |    |_  |/  ___|/  __ \| |   |_   _|
|  \| |  ___  ___ | |_     | |\ `--. | /  \/| |     | |
| . ` | / _ \/ __|| __|    | | `--. \| |    | |     | |
| |\  ||  __/\__ \| |_ /\__/ //\__/ /| \__/\| |_____| |_
```

```
\_| \_/ \___||___/ \__|\____/ \____/  \____/\_____/\___/

[System Information]
OS Version : macOS High Sierra
NodeJS Version : v16.18.0
[Nest Information]
microservices version : 10.0.0
websockets version : 10.0.0
testing version : 10.0.0
common version : 10.0.0
core version : 10.0.0
```