Nx Monorepo build system

CONCEPTS

A project has targets (build, test, ...). Nx runs a target, replays cached output when inputs are unchanged, and runs dependencies first.

Project graph projects + deps
↓ target deps
Task graph what to run
↓ hash inputs
Cache hit: replay
↓ miss: run
Run parallel

Without a global install use npx nx, pnpm nx or yarn nx; the workspace version is used.

RUN TASKS

`nx run <project>:<target>`
Run one target

`nx <target> <project>`
Shorthand for run

`nx run <p>:<t>:<config>`
Use a configuration

`nx <target> <project> -c <config>`
Set configuration by flag

`nx run-many -t <target>`
Run a target everywhere

`nx run-many -t <t1> <t2>`
Run several targets

`nx run-many -t <target> -p <a> <b>`
Limit to projects

`nx run-many -t <target> -p "tag:<tag>"`
Select projects by tag

`nx run-many -t <target> --exclude <p>`
Skip projects

`nx affected -t <target>`
Only affected projects

`nx affected -t <target> --base=<ref> --head=<ref>`
Compare two refs

`nx affected -t <target> --uncommitted`
Include local edits

RUN TASKS (cont.)

`--parallel=<n>`
Max concurrent tasks (default 3)

`--parallel=false`
Run one task at a time

`--nxBail`
Stop after first failure

INSPECT

`nx graph`
Open the project graph

`nx graph --focus=<project>`
One project and neighbours

`nx show projects`
List projects

`nx show projects --withTarget=<t>`
Projects with a target

`nx show projects --affected`
Only affected ones

`nx show project <project>`
Resolved project config

GENERATE AND SET UP

`nx g <plugin>:<generator> <name>`
Run a generator

`nx g <plugin>:<generator> --dry-run`
Preview file changes

`nx g @nx/js:library <name>`
New library

`nx add <package>`
Install and set up a plugin

`nx init`
Add Nx to an existing repo

`nx migrate latest`
Plan an upgrade

`nx migrate --runMigrations`
Apply migrations.json

CACHE AND DAEMON

`nx reset`
Clear cache, daemon, graph data

`nx reset --onlyDaemon`
Restart the daemon

CACHE AND DAEMON (cont.)

`--skipNxCache`
Bypass local cache for a run

A cache hit replays terminal output and restores outputs. Stale results usually mean inputs are missing from the hash.

CONFIG FILES

`project.json: targets`
Per-project task definitions

`targets.<t>.executor`
What the task runs

`targets.<t>.dependsOn`
Tasks that run first

`"^build" in dependsOn`
Same target in dependencies

`targets.<t>.inputs`
Files and vars that affect hash

`targets.<t>.outputs`
Paths to cache

`targets.<t>.cache: true`
Make a target cacheable

`nx.json: targetDefaults`
Shared target settings

`nx.json: plugins`
Inferred targets from tools

`nx.json: defaultBase`
Base branch for affected

RECIPES

nx.json defaults

```
{
  "defaultBase": "main",
  "targetDefaults": {
    "build": {
      "dependsOn": ["^build"],
      "cache": true,
      "outputs": ["{projectRoot}/
dist"]
    },
    "test": { "cache": true }
  }
}
```

CI: only what a PR touched

```
nx affected -t lint test build --
base=origin/main
# shallow clones hide history:
# fetch full history (fetch-depth:
0)
```