Nx Monorepo build system

START HERE

npx nx *<target> <project>*
Run one task

nx **reset**
Clear cache and daemon

nx **run-many -t build**
Run in all projects

nx *<target> <project>* **--skipNxCache**
Bypass the cache

nx **affected -t test**
Only what changed

nx **graph**
See the project graph

nx **show projects**
List projects

nx **g** *<plugin>:<generator>*
Scaffold code

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

CONCEPTS

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

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

Task flags not known to Nx can be passed after the project: nx test web --watch.

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

RUN TASKS (cont.)

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

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

--parallel=false
Run one task at a time

--nxBail
Stop after first failure

--outputStyle=*<style>*
tui, stream, static, summary

--graph
Show the task graph, no run

nx **watch --all --** *<cmd>*
Rerun a command on change

INSPECT

nx **graph**
Open the project graph

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

nx **graph --affected**
Highlight affected

INSPECT (cont.)

nx **graph --view=tasks --targets=***<t>*
Task graph view

nx **graph --file=***<out>***.json**
Write graph to file

nx **show projects**
List projects

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

nx **show projects --type=***<app|lib|e2e>*
Filter by type

nx **show projects --affected**
Only affected ones

nx **show project** *<project>*
Resolved project config

nx **show target inputs** *<project>:<t>*
What feeds the hash

nx **show target outputs** *<project>:<t>*
What gets cached

GENERATE AND SET UP

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

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

GENERATE AND SET UP (cont.)

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 **sync**
Run sync generators

nx **sync:check**
Fail if out of sync (CI)

nx **format:write**
Format affected files

nx **format:check**
Check formatting (CI)

nx **migrate latest**
Plan an upgrade

nx **migrate --runMigrations**
Apply migrations.json

nx **migrate latest --createCommits**
Commit per migration

CACHE AND DAEMON

nx **reset**
Clear cache, daemon, graph data

nx **reset --onlyCache**
Clear only local cache

nx **reset --onlyDaemon**
Restart the daemon

CACHE AND DAEMON (cont.)

--skipNxCache
Bypass local cache for a run

--skipRemoteCache
Bypass remote cache

NX_DAEMON=false
Disable daemon 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

project.json: tags
Labels for tag: selectors

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

targets.*<t>***.command**
Plain shell command

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

CONFIG FILES (cont.)

targets.*<t>***.configurations**
Named option sets

nx.json: targetDefaults
Shared target settings

nx.json: namedInputs
Reusable input groups

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)
```

Upgrade Nx

```
nx migrate latest
npm install
nx migrate --runMigrations
# review diff, then delete
migrations.json
```