Angular CLI

Angular workspace tool

START HERE

`ng new <name>`
Create a workspace

`ng build`
Compile to dist/

`ng serve -o`
Run dev server, open browser

`ng test`
Run unit tests

`ng generate component <name>`
Add a component

`ng add <package>`
Install with setup

`ng generate service <name>`
Add a service

`ng update`
Check for upgrades

angular.json projects, targets → `ng <command>` → **Builder** build, serve, test → `runs` → **dist/** output or dev server

CONCEPTS

Run ng from anywhere inside a workspace; it finds angular.json upwards. ng new is the only command that works outside one.

Every command is a target in angular.json: ng run <project>:<target> is the generic form behind them.

Flags are dash-case on the command line (--output-path) but camelCase in angular.json (outputPath).

WORKSPACE

`ng new <name>`
Create a new workspace and app
Prompts for style, SSR and more; add --defaults to skip prompts.

`ng new <name> --routing --style=scss`
Create with routing and SCSS

`ng version`
Show CLI, Angular and package versions

`ng add <package>`
Install a package and run its setup
Works only for packages that ship an ng-add schematic.

`ng new <name> --skip-install --skip-git`
Create without install or git init

`ng new <name> --package-manager=pnpm`
Pick the package manager

`ng new <name> --ssr`
Create with server-side rendering

`ng new <name> --prefix=<prefix>`
Set selector prefix (default app)

`ng completion script`
Print shell completion script
Add source <(ng completion script) to your shell rc file.

SERVE AND BUILD

`ng serve`
Start the dev server (port 4200)

`ng serve -o`
Start and open the browser

SERVE AND BUILD (cont.)

`ng serve --port <n>`
Listen on another port

`ng build`
Build the app into dist/

`ng build -c production`
Build with a named configuration

`ng serve -c <config>`
Serve using a named configuration

`ng serve --host 0.0.0.0`
Listen on all interfaces
If other hosts are refused, look at --allowed-hosts.

`ng serve --proxy-config <file>`
Proxy API calls to a backend

`ng serve --ssl`
Serve over HTTPS

`ng build --watch`
Rebuild on file changes

`ng build --base-href <path>`
Set the base URL for deployment

`ng build --output-path <dir>`
Write output elsewhere

`ng build --stats-json`
Write stats for bundle analysis

`ng extract-i18n`
Extract translatable messages

GENERATE

`ng generate component <name>`
Create a component
ng g is the short form of ng generate.

`ng generate service <name>`
Create a service

`ng generate directive <name>`
Create a directive

`ng generate pipe <name>`
Create a pipe

`ng generate guard <name>`
Create a route guard

`ng generate interface <name>`
Create an interface

`ng generate component <path/name>`
Create inside a subfolder
The path is relative to src/app.

GENERATE (cont.)

`ng generate component <name> --dry-run`
Preview files, write nothing

`ng generate component <name> --skip-tests`
Skip the .spec.ts file

`ng generate component <name> --inline-template --inline-style`
Single-file component

`ng generate component <name> --change-detection Eager`
Use default change detection
OnPush is the default for new components.

`ng generate interceptor <name>`
Create an HTTP interceptor

`ng generate library <name>`
Add a library project

`ng generate application <name>`
Add another app to the workspace

TEST AND LINT

`ng test`
Run unit tests
The default runner is vitest; tests run in Node with jsdom unless --browsers is set.

`ng test --no-watch`
Run once and exit

`ng test --coverage`
Collect a coverage report

`ng test --filter <regex>`
Run tests whose name matches

`ng test --browsers <name>`
Run in a real browser

`ng test --include <glob>`
Limit to matching spec files

`ng test --runner karma`
Use Karma instead of Vitest

`ng test --reporters <list>`
Choose output reporters

`ng lint`
Run the configured linter
Needs a linter first: ng add angular-eslint.

`ng add angular-eslint`
Set up ESLint for lint

`ng e2e`
Build, serve and run e2e tests
Needs an e2e package added with ng add first.

MAINTAIN

`ng update`
List packages that can be updated

`ng update @angular/core @angular/cli`
Update Angular and run migrations
Commit first: it refuses to run on a dirty git tree.

`ng update --create-commits`
Commit each update and migration

`ng update <package> --migrate-only --from <ver>`
Re-run migrations only

`ng update --force`
Ignore peer dependency conflicts

`ng config <path>`
Read a value from angular.json

`ng config <path> <value>`
Set a value in angular.json

`ng cache info`
Show cache location and size

`ng cache clean`
Delete the build cache

`ng run <project>:<target>`
Run any target from angular.json

A stale cache lives in .angular/cache; clean it when builds behave oddly.

RECIPES

New app with SCSS and routing, then serve

```
ng new shop --routing --
style=scss
cd shop
ng serve -o
```

Feature skeleton

```
ng generate component products/
list --dry-run
ng generate component products/
list
ng generate service products/
products
```

Safe upgrade

```
git commit -am "wip"
ng update
ng update @angular/core
@angular/cli --create-commits
ng build && ng test --no-watch
```

Dev server with API proxy

```
# proxy.conf.json
{ "/api": { "target": "http://
localhost:3000" } }
ng serve --proxy-config
proxy.conf.json
```