Scalable enterprise architectures with Nrwl NX

Dmitriy Stepanenko, Software Engineer

Valor Software

What is NX

"Nx is a smart, fast and extensible build system with first class monorepo support and powerful integrations.".

Nx Docs

A workspace tooling system..

To:

- architect
- build
- test
- deploy

For:

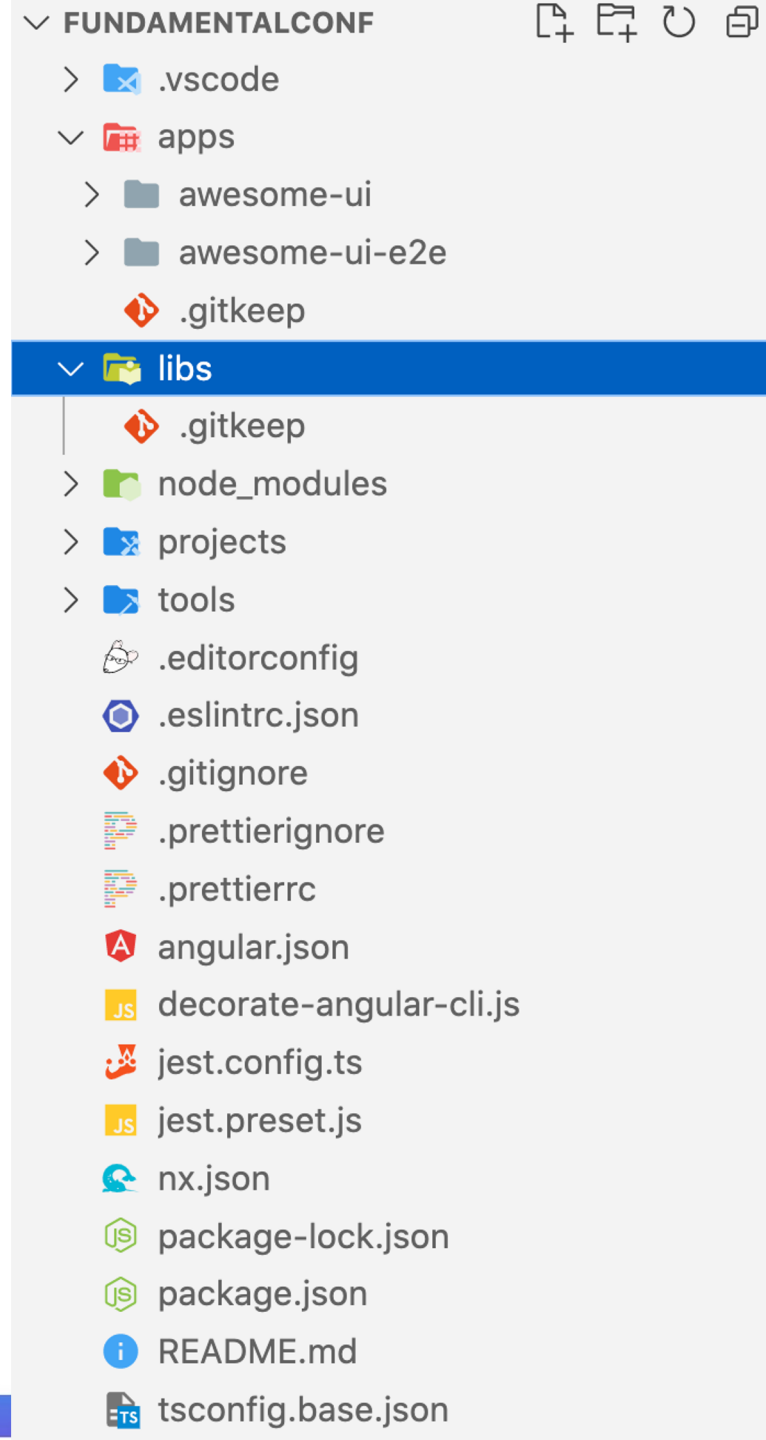
- VanillaJs
- Frontend frameworks (e.g. Angular/React/NextJs)
- Backend frameworks (e.g. Express/NestJs)

That is targeted for:

- multi-project monorepositors
- single projects within the repo

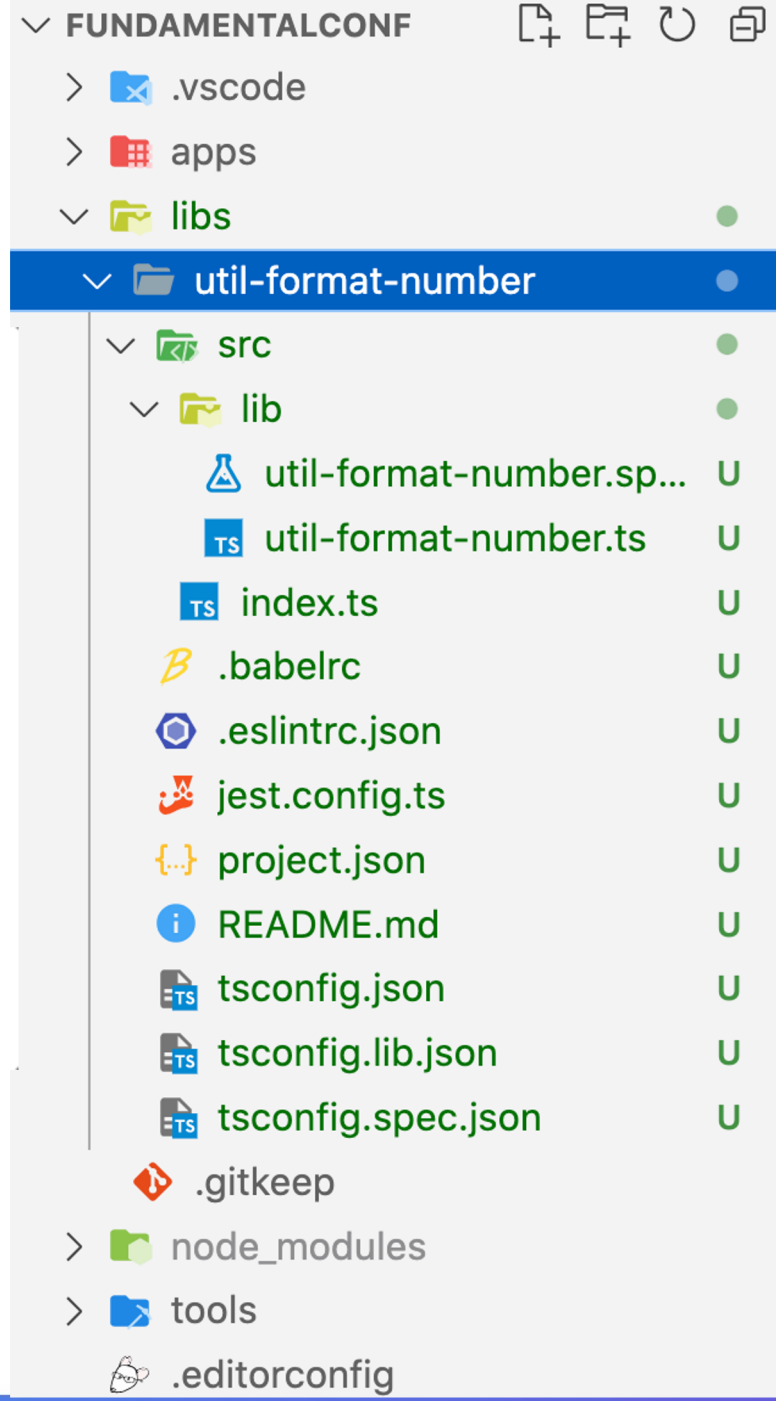
Getting started

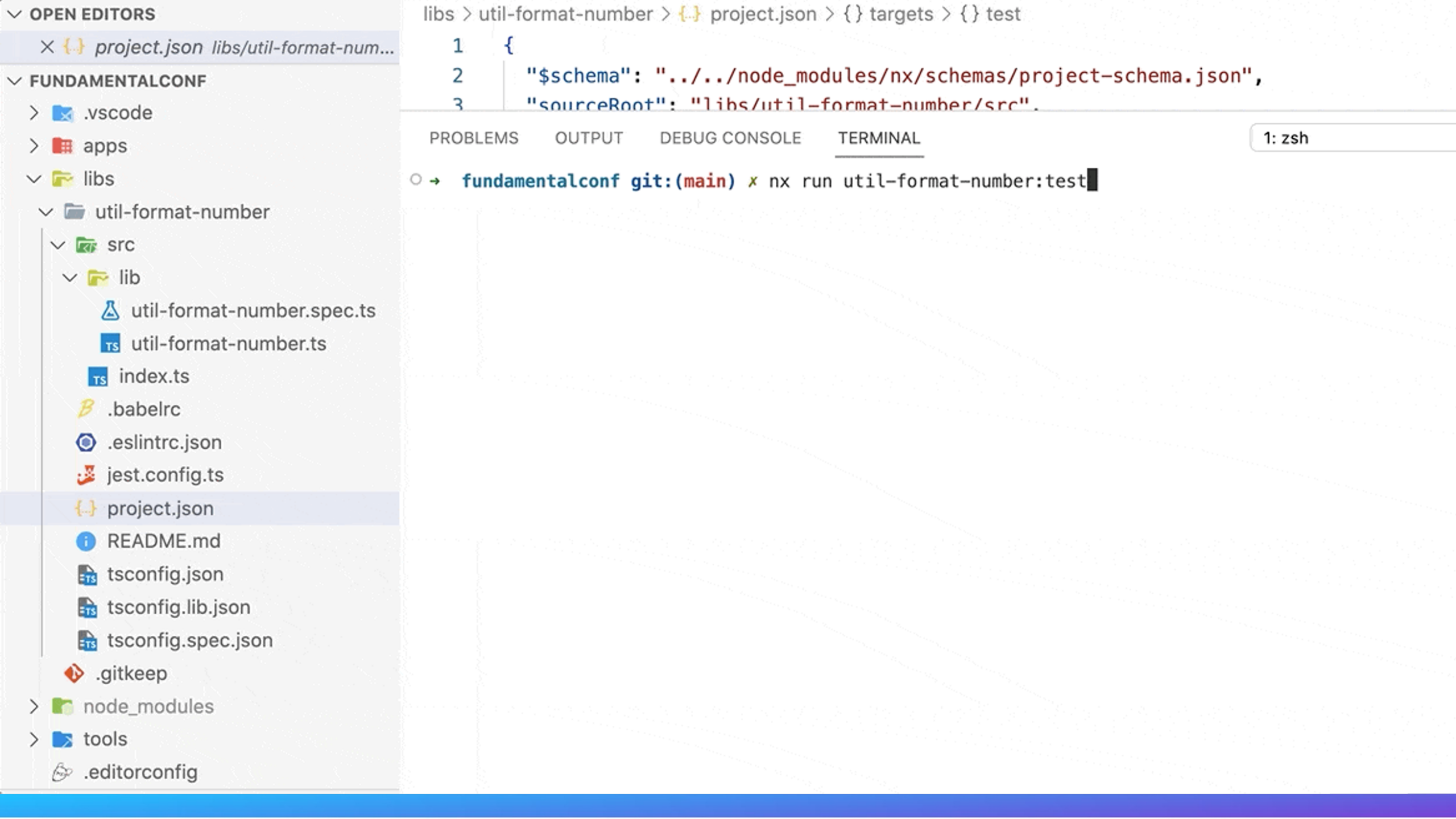
```
➜  projects npx create-nx-workspace fundamentalconf
? What to create in the new workspace …
✔ What to create in the new workspace · angular
✔ Application name                     · awesome-ui
✔ Default stylesheet format            · scss
✔ Enable distributed caching to make your CI faster · Yes
```

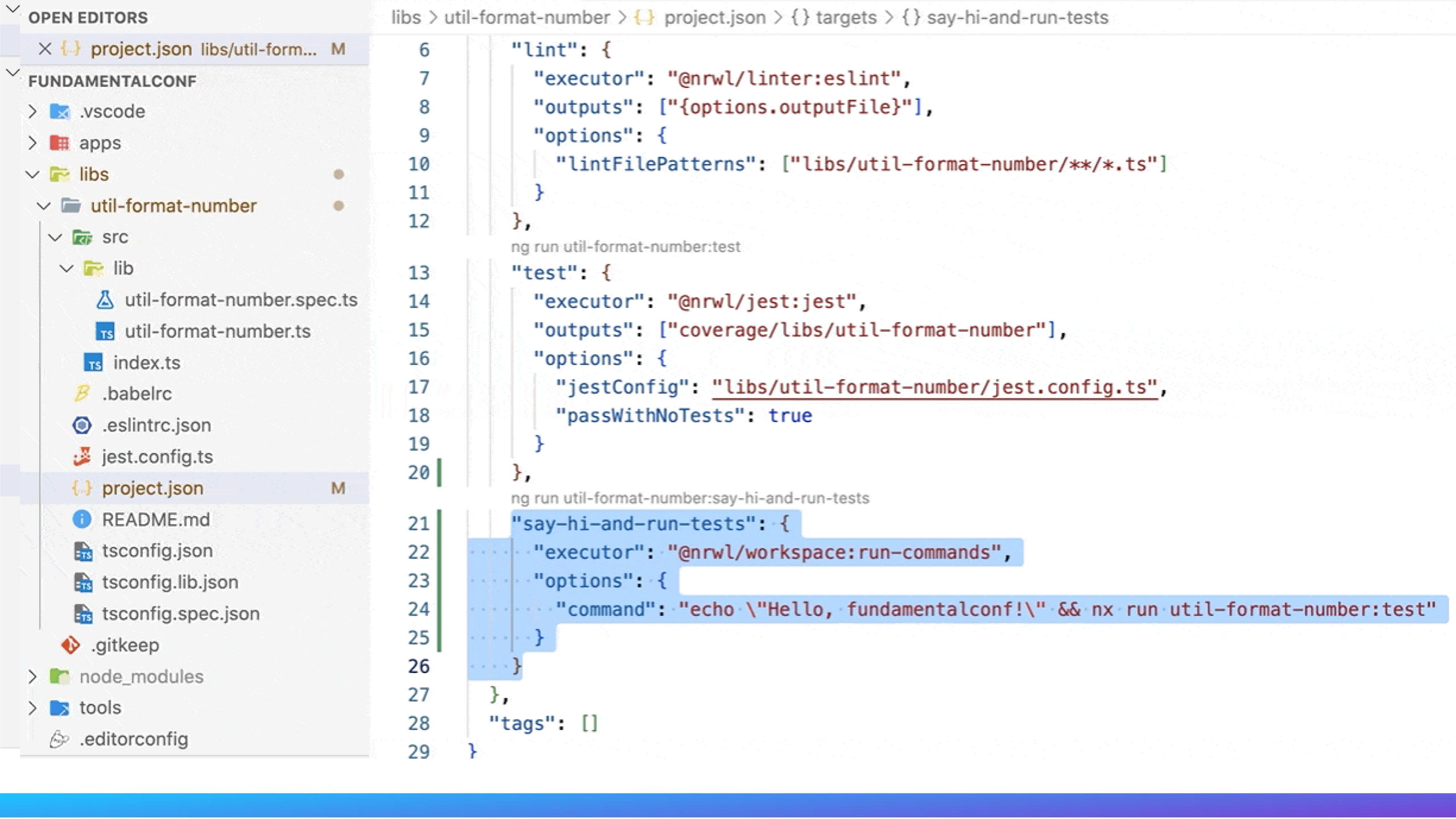


Using libraries

```
→ fundamentalconf git:(main) npx nx generate @nrwl/workspace:library util-format-number
```




OPEN EDITORS
project.json libs/util-format-num...
FUNDAMENTALCONF
.vscode
apps
libs
util-format-number
src
lib
util-format-number.spec.ts
util-format-number.ts
index.ts
.babelrc
.eslintrc.json
jest.config.ts
project.json
README.md
tsconfig.json
tsconfig.lib.json
tsconfig.spec.json
.gitkeep
node_modules
tools
.editorconfig
libs > util-format-number > project.json > { } targets > { } test
1 {
2 "$schema": "../../node_modules/nx/schemas/project-schema.json",
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: zsh
→ fundamentalconf git:(main) ✗ nx run util-format-number:test


OPEN EDITORS
project.json libs/util-form... M
FUNDAMENTALCONF
.vscode
apps
libs
util-format-number
src
lib
util-format-number.spec.ts
util-format-number.ts
index.ts
.babelrc
.eslintrc.json
jest.config.ts
project.json M
README.md
tsconfig.json
tsconfig.lib.json
tsconfig.spec.json
.gitkeep
node_modules
tools
.editorconfig
libs > util-format-number > project.json > {} targets > {} say-hi-and-run-tests
"lint": {
"executor": "@nrwl/linter:eslint",
"outputs": ["{options.outputFile}"],
"options": {
"lintFilePatterns": ["libs/util-format-number/**/*.ts"]
}
},
ng run util-format-number:test
"test": {
"executor": "@nrwl/jest:jest",
"outputs": ["coverage/libs/util-format-number"],
"options": {
"jestConfig": "libs/util-format-number/jest.config.ts",
"passWithNoTests": true
}
},
ng run util-format-number:say-hi-and-run-tests
"say-hi-and-run-tests": {
"executor": "@nrwl/workspace:run-commands",
"options": {
"command": "echo \"Hello, fundamentalconf!\" && nx run util-format-number:test"
}
}
},
"tags": []
}

Caching task results

"It's costly to rebuild and retest the same code over and over again. Nx uses a computation cache to never rebuild the same code twice."

Nx Docs

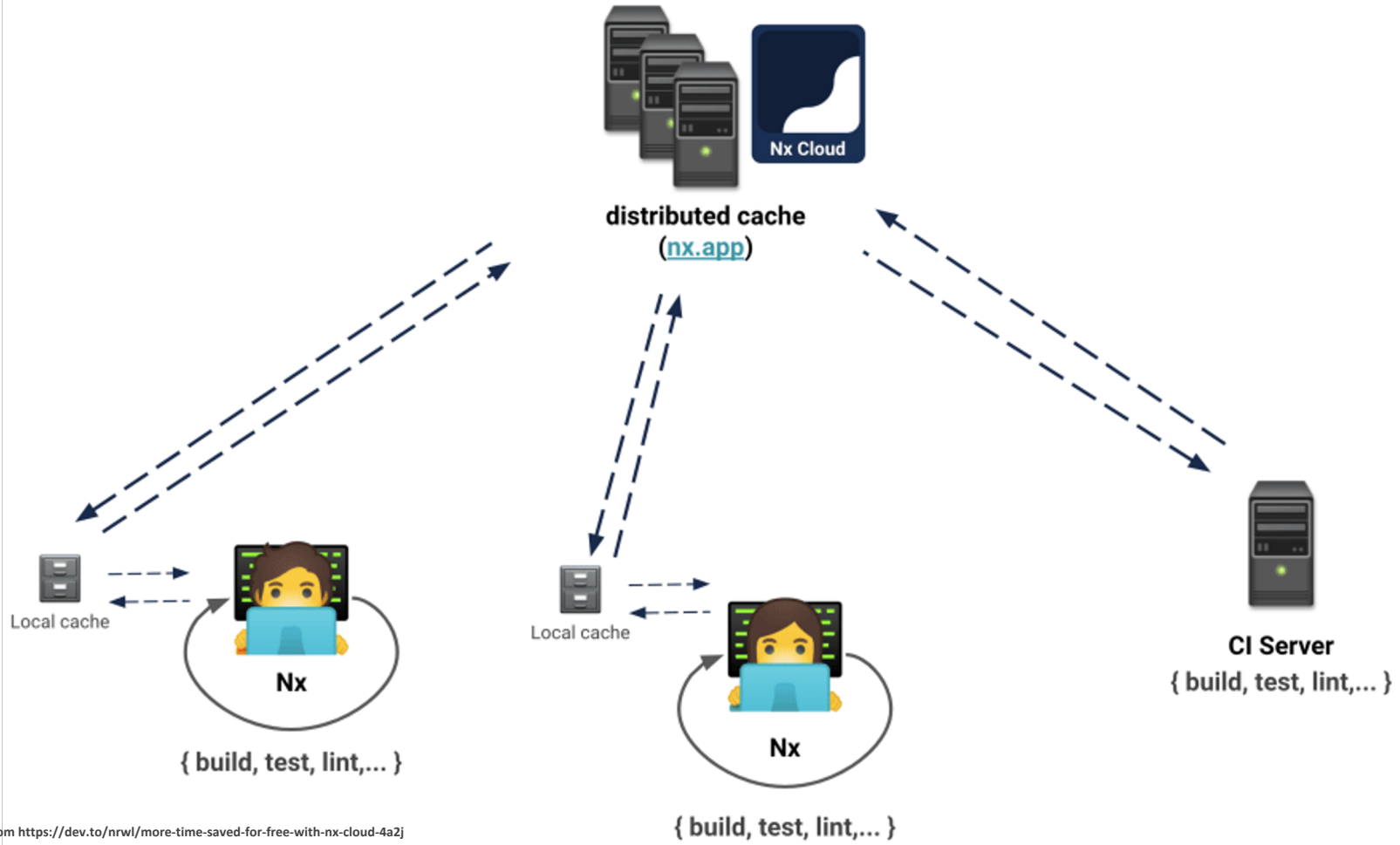
How it works:

- Compares hashes with saved ones and instantly returns the result if match found
- Works for any targets (library commands) using Nx Cli

Computation hash includes:

- All the source files of the provided project and its dependencies
- Relevant global configuration
- Versions of external dependencies
- Runtime values provisioned by the user such as the version of Node, bash commands or scripts
- CLI Command flags

Caching task results, Nx Cloud



Attempt to retrieve cache locally

↓

Attempt to retrieve from Nx Cloud

↓

No cache found, execute the task

Taken from https://dev.to/nrwl/more-time-saved-for-free-with-nx-cloud-4a2j

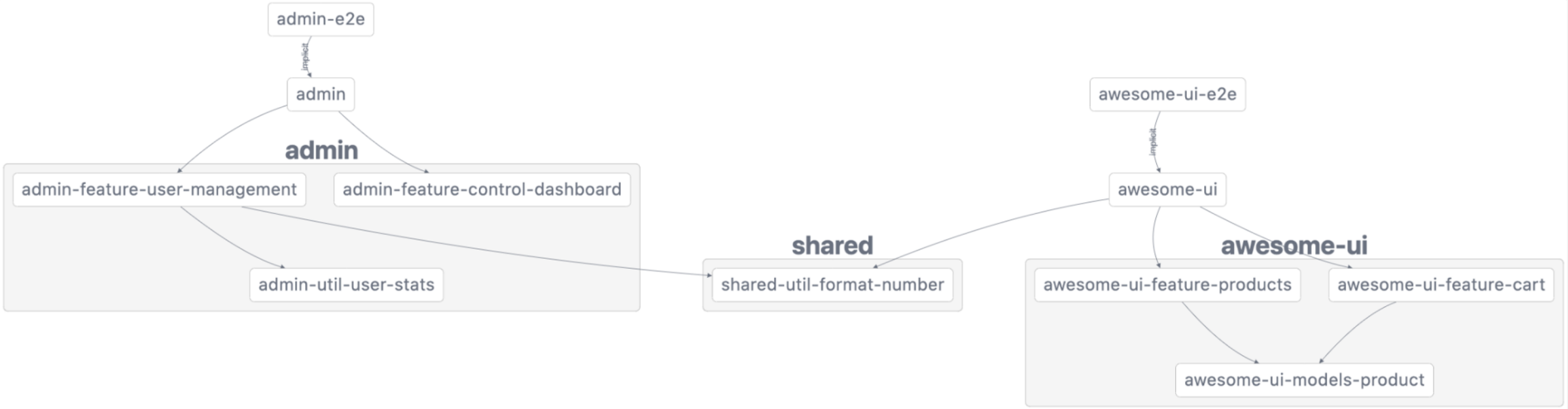
Dependency graph

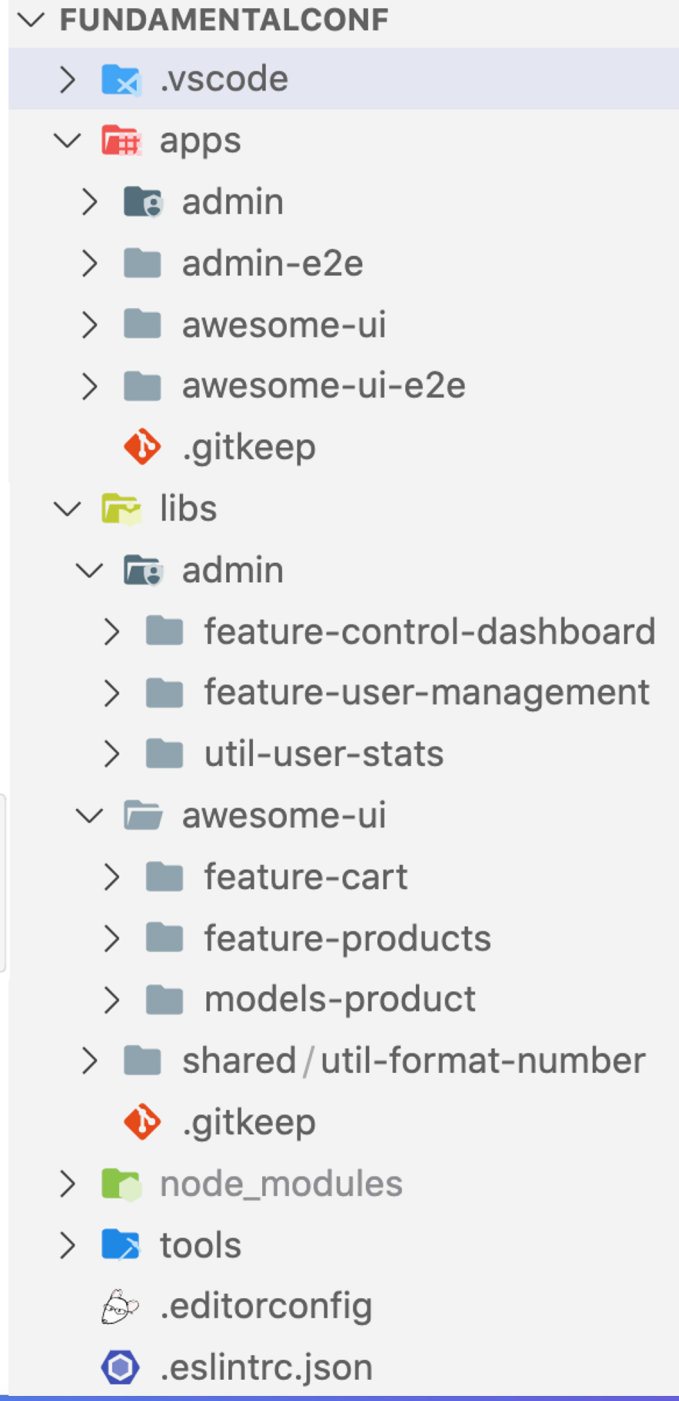
```
➜  fundamentalconf git:(main) nx dep-graph
```

Dependency graph

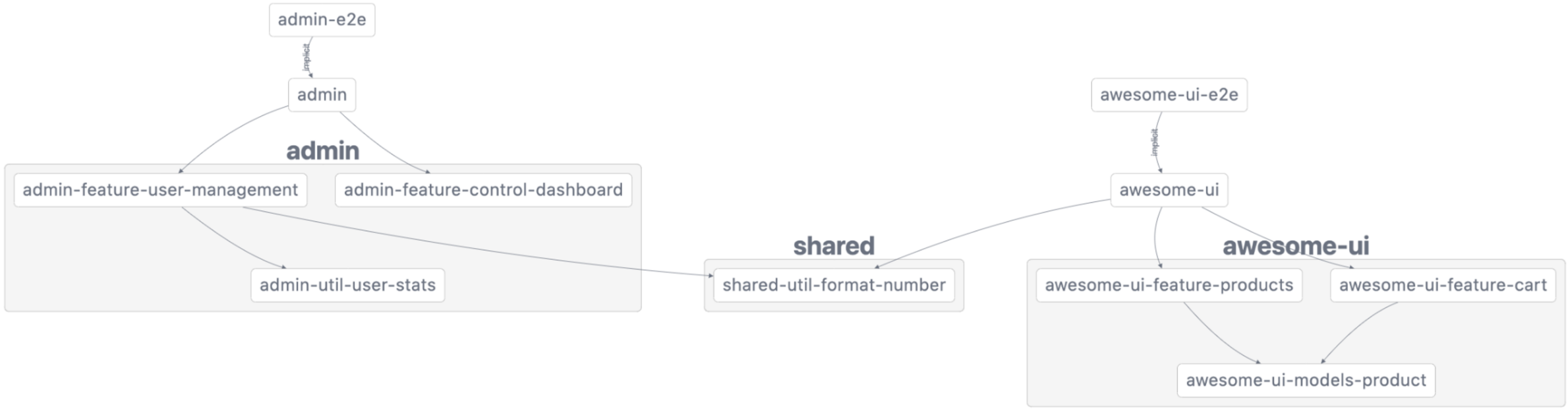
After generating a few libraries project structure is as follows

```
➜  fundamentalconf git:(main) nx dep-graph
```





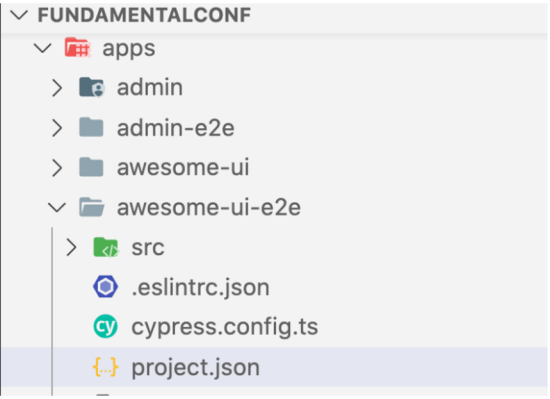
Dependency graph



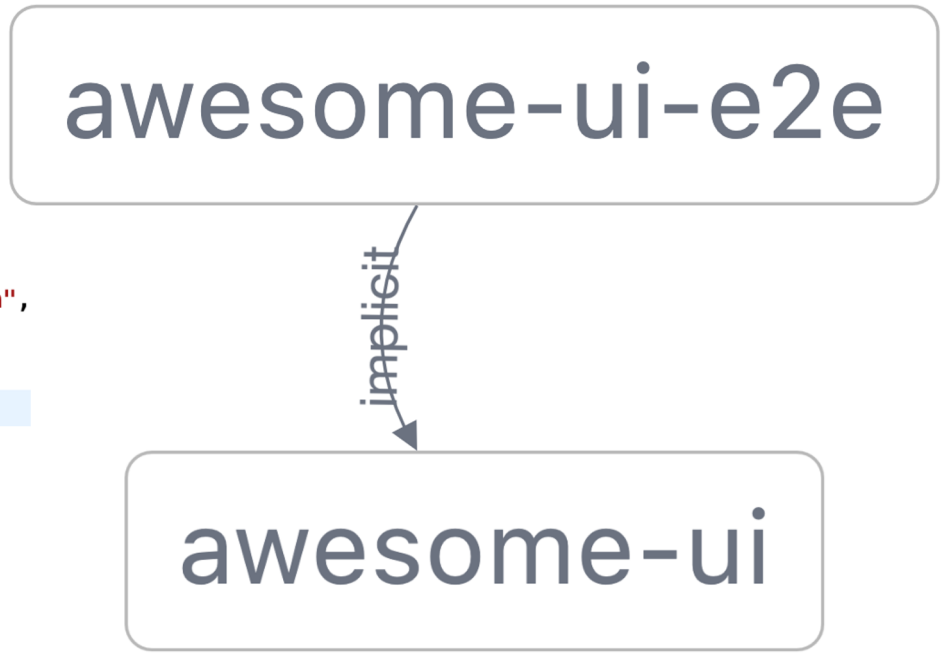
Dependencies are determined by:

- Typescript import statements
- *“implicitDependencies”* - manually configured references in the library configuration file.

Dependency graph




```
1  {
2    "$schema": "../../node_modules/nx/schemas/project-schema.json",
3    "sourceRoot": "apps/awesome-ui-e2e/src",
4    "projectType": "application",
5    "targets": {…
26   },
27   "tags": [],
28   "implicitDependencies": ["awesome-ui"]
29 }
30
```



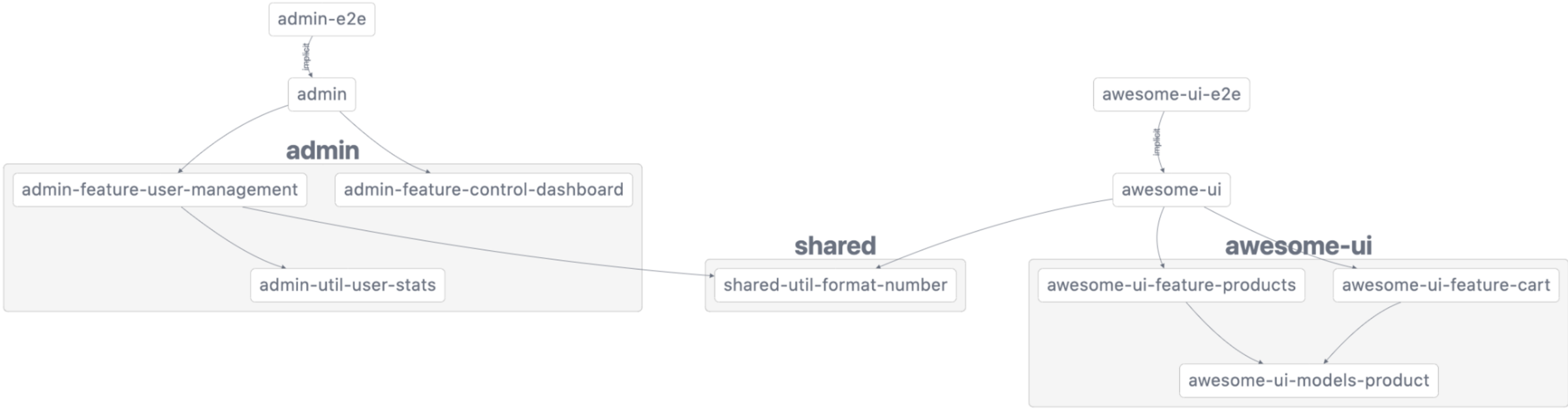
Dependencies are determined by:

- Typescript import statements
- *"implicitDependencies"* - manually configured references in the library configuration file.

Affected

```
→  fundamentalconf git:(main) ✗ nx affected:dep-graph
```

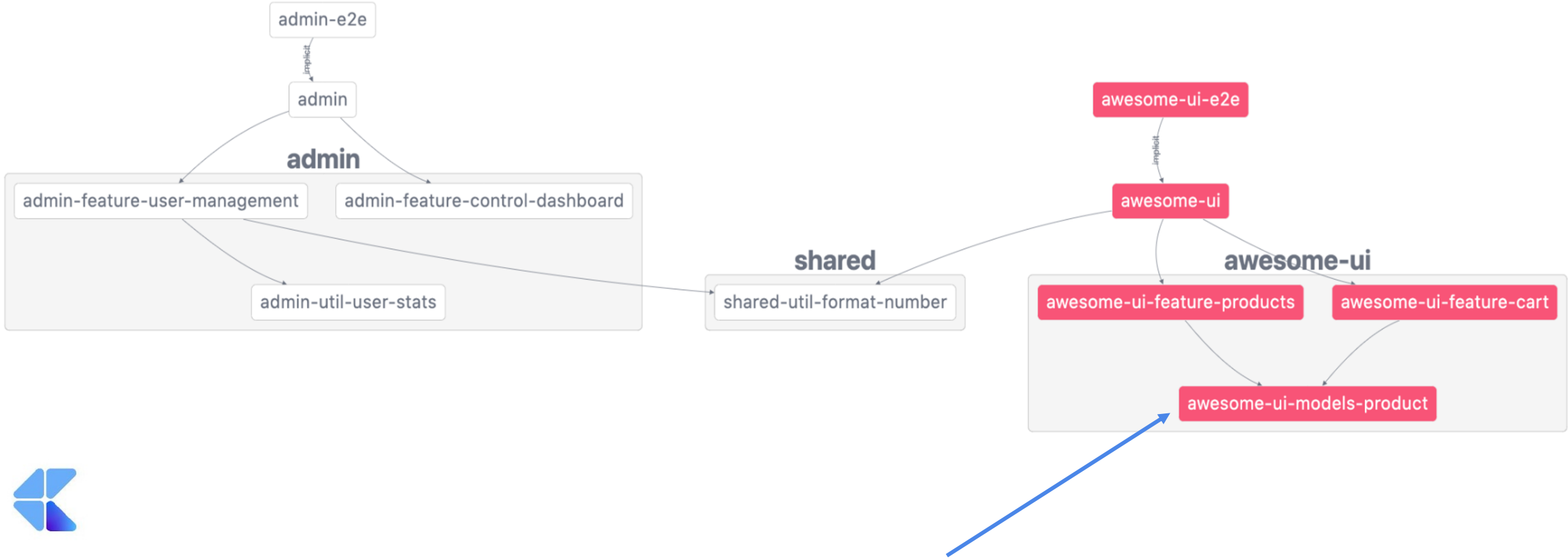
No files changed



Affected

```
→  fundamentalconf git:(main) ✗ nx affected:dep-graph
```

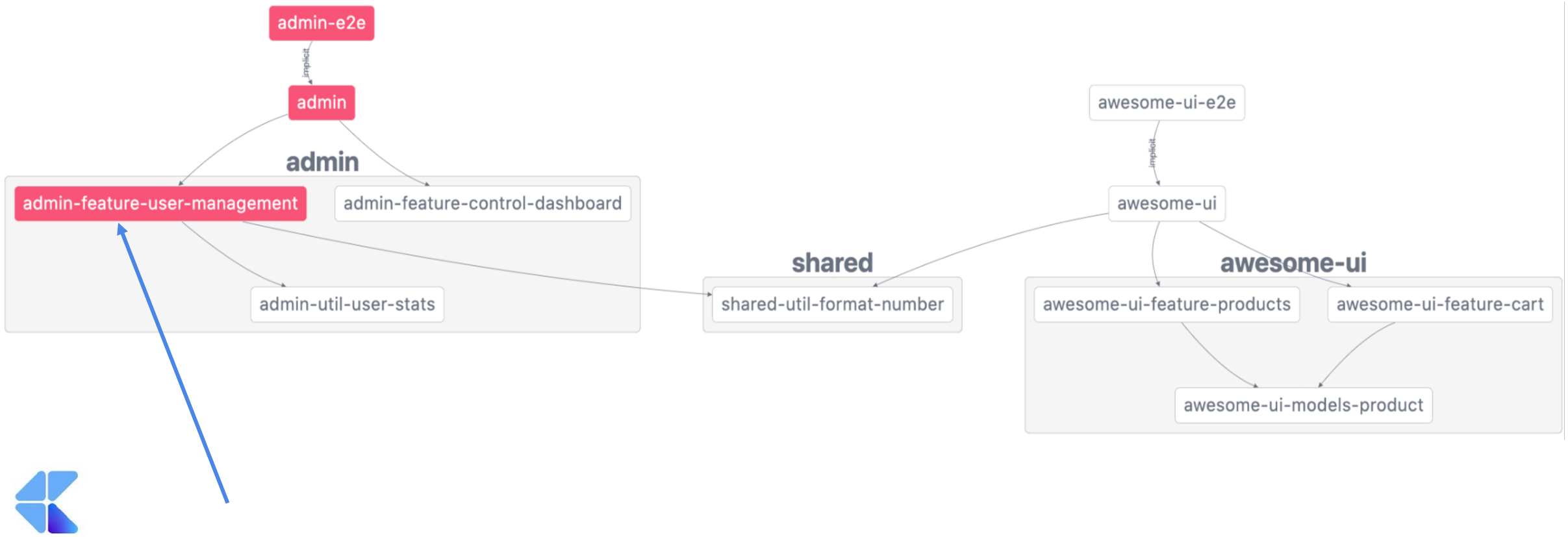
Change in: libs/awesome-ui/models-product/src/lib/product.model.ts



Affected

```
→  fundamentalconf git:(main) ✗ nx affected:dep-graph
```

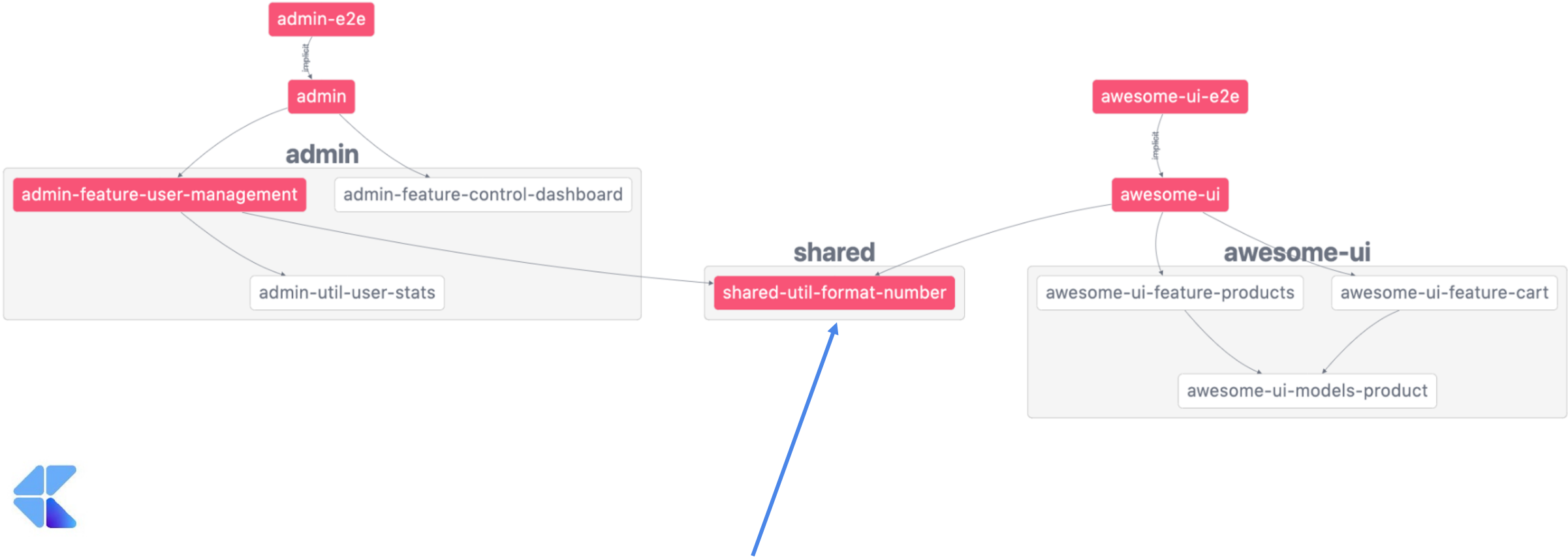
Change in: libs/admin/feature-user-management/src/lib/admin-feature-user-management.module.ts



Affected

```
→  fundamentalconf git:(main) ✗ nx affected:dep-graph
```

Change in: libs/shared/util-format-number/src/lib/util-format-number.ts

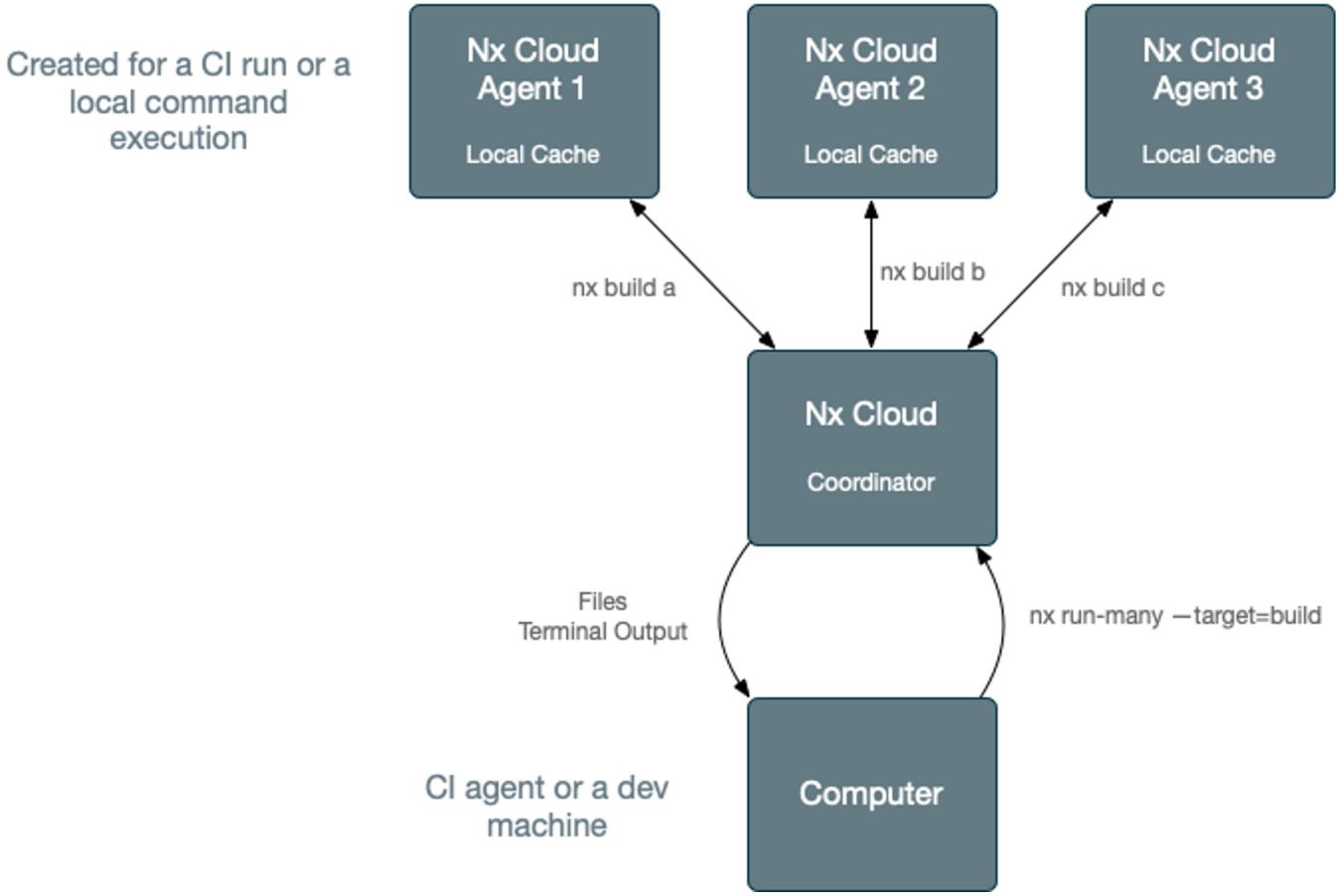


Affected

```
● → fundamentalconf git:(main) ✗ nx affected --target=test --parallel=3

  > NX   Affected criteria defaulted to --base=main --head=HEAD


    ✔  nx run admin-feature-user-management:test (4s)
    ✔  nx run admin:test (8s)
    ✔  nx run awesome-ui:test (8s)
    ✔  nx run shared-util-format-number:test (6s)

————————————————————————————————————————————————————————————————

  > NX   Successfully ran target test for 4 projects (10s)

    See Nx Cloud run details at https://nx.app/runs/c0JIi57eiD

○ → fundamentalconf git:(main) ✗
```

- running actions only for targets, affected by the change
- significantly shortening the time needed for CI to run
- targets can run in parallel (default is 3 parallel executions)
- comparison is done between two Git revisions (by default between **main** branch and **HEAD**)

Distributed Task Execution

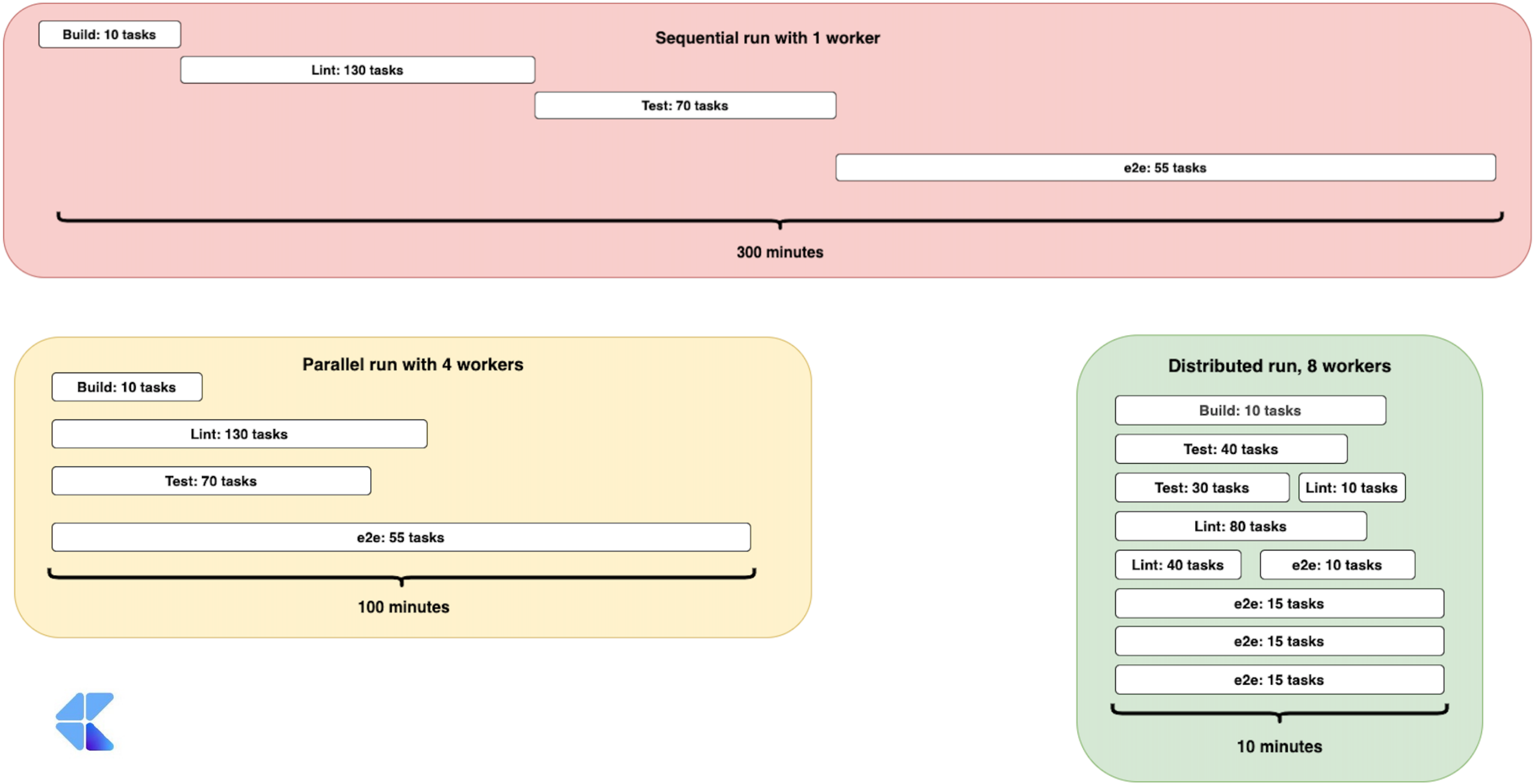


Taken from https://blog.nrwl.io/distributing-ci-binning-and-distributed-task-execution-632fe31a8953

How it works:

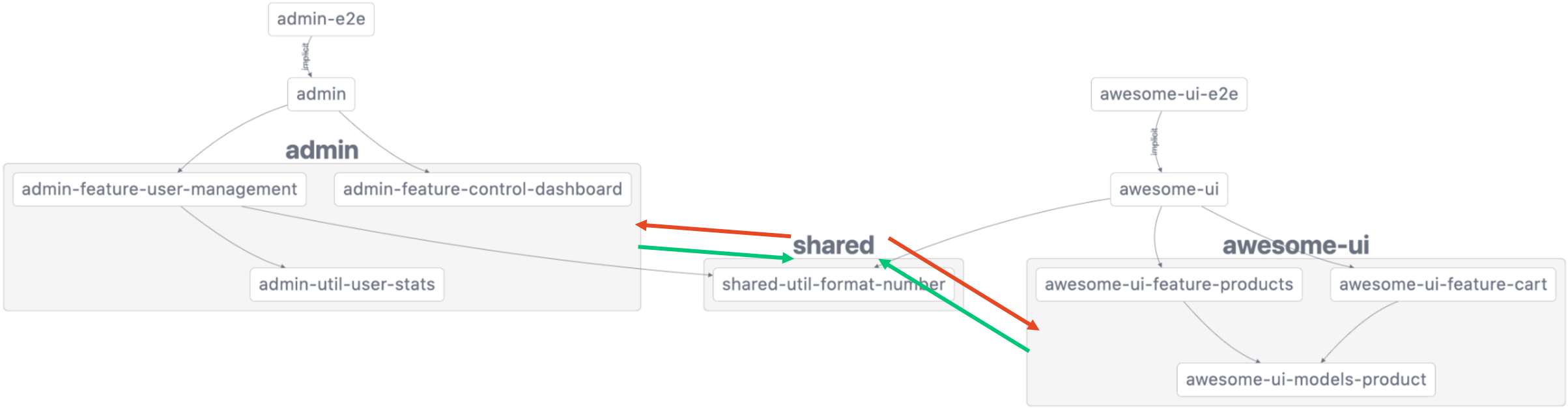
1. **You** define:
   a. What tasks to run (lint, test, build)
   b. How many agents to use
2. **Nx Cloud Coordinator** analizes what tasks should be run and in what order
3. **Nx Cloud Coordinator** tracks what agent has finished current task and sends the next one

And keep in mind, everything here is cached using Nx Cloud!



Only for visualization. On practise, with DTE all workers will run all tasks, not separated by type as it's shown here

Structuring your Nx workspace



In order to set up tags:

- add required relations into .eslintrc.json
- mark each lib's project json with that tag

Each lib's project.json :

project.json

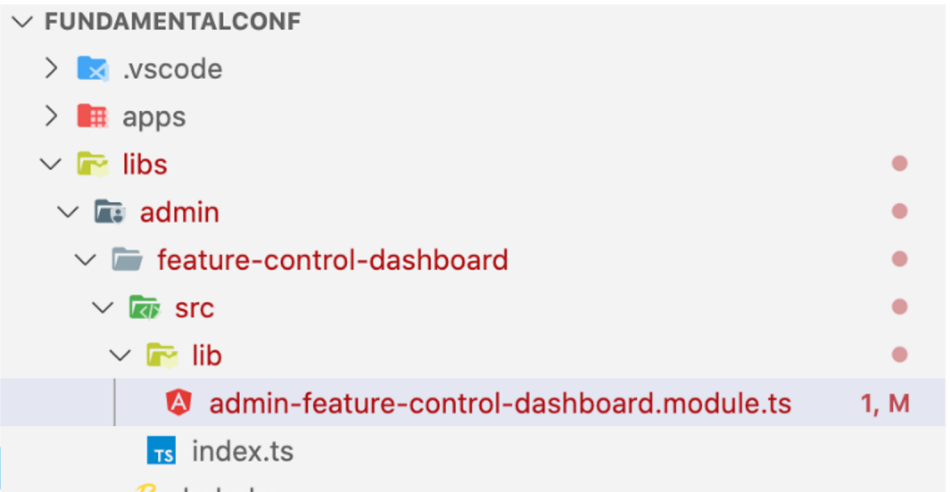
libs > admin > feature-control-dashboard > {..} project.json > [ ] tags

```
1   {
2     "$schema": "../../../node_modules/nx/schemas/project-schema.json",
3     "sourceRoot": "libs/admin/feature-control-dashboard/src",
4     "projectType": "library",
5   > "targets": {…
21    },
22    "tags": ["admin"]
23  }
24
```

.eslintrc.json :

```
8       "rules": {
9         "@nrwl/nx/enforce-module-boundaries": [
10          "error",
11          {
12            "enforceBuildableLibDependency": true,
13            "allow": [],
14            "depConstraints": [
15              {
16                "sourceTag": "awesome-ui",
17                "onlyDependOnLibsWithTags": ["awesome-ui", "shared"]
18              },
19              {
20                "sourceTag": "admin",
21                "onlyDependOnLibsWithTags": ["admin", "shared"]
22              },
23              {
24                "sourceTag": "shared",
25                "onlyDependOnLibsWithTags": ["shared"]
26              }
27            ]
28          }
29        ]
```

Attempting to import library from "awesome-ui" into "admin" library:



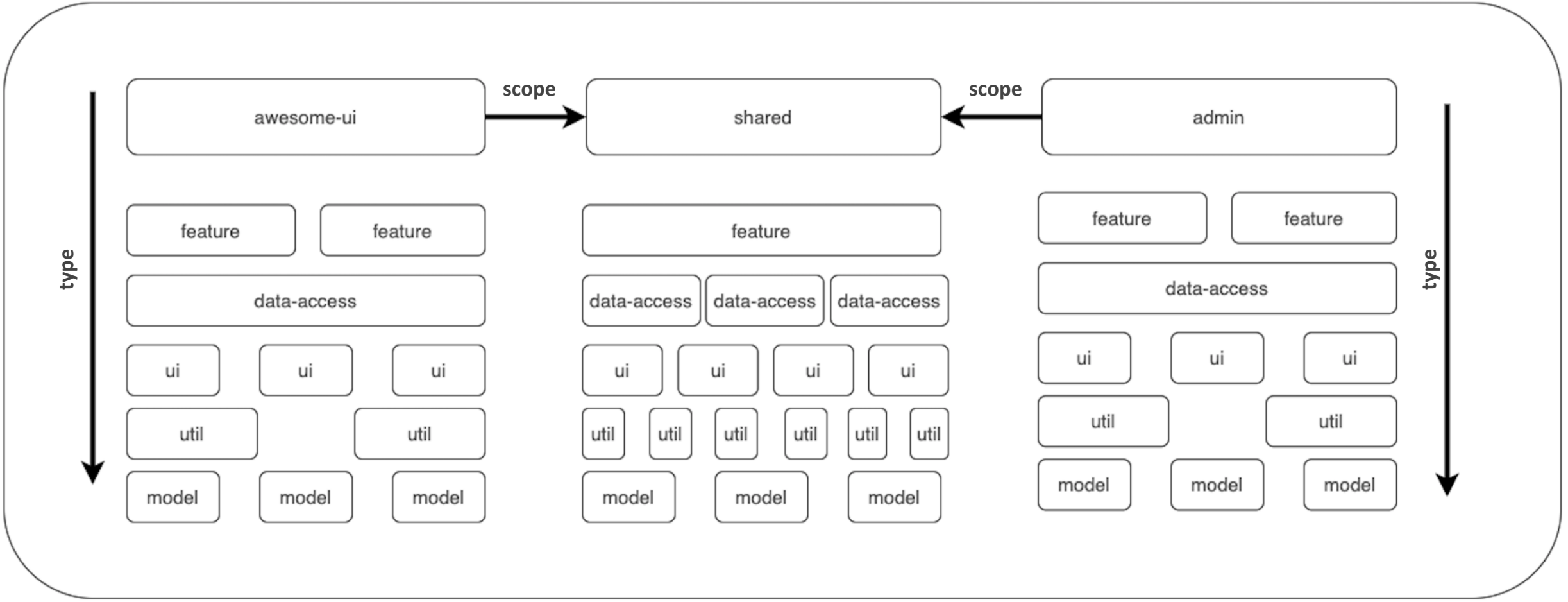

```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { AwesomeUiFeatureCartModule } from '@fundamentalconf/awesome-ui/feature-cart';
4
5     (alias) class AwesomeUiFeatureCartModule
6     import AwesomeUiFeatureCartModule
7
8     A project tagged with "admin" can only depend on libs tagged with "admin",
      "shared" eslint(@nrwl/nx/enforce-module-boundaries)
9
      View Problem (⌘K N)    Quick Fix... (⌘.)
10  })
11  export class AdminFeatureControlDashboardModule {}
```

- **feature** - libraries that implement smart UI (with access to data sources)
- **data-access** - code for interacting with a back-end system or state management
- **ui** (frontend-only) - contains presentational components (also called "dumb" components).
- **util** - low-level utilities used by many libraries and applications.
- **model** - interfaces reused between libraries and applications.

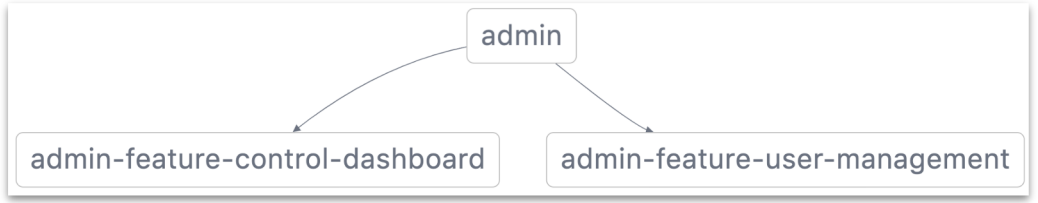
project.json M

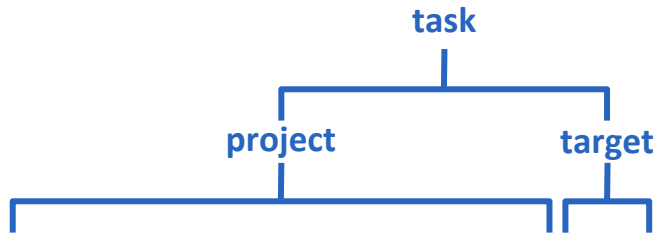
libs > admin > feature-control-dashboard > project.json > [ ] tags

```
1  {
2    "$schema": "../../../node_modules/nx/schemas/project-sc
3    "sourceRoot": "libs/admin/feature-control-dashboard/src
4    "projectType": "library",
5  > "targets": {…
21   },
22   "tags": ["scope:admin", "type:feature"]
23 }
```



Summing up

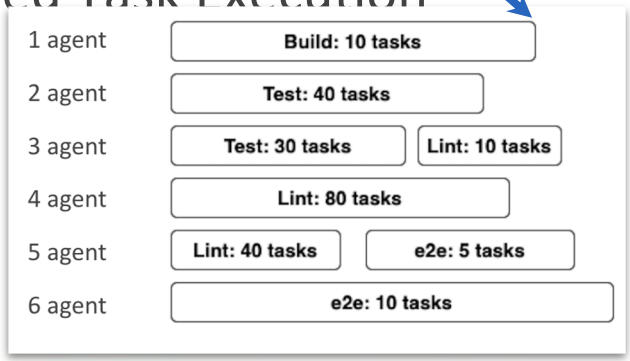


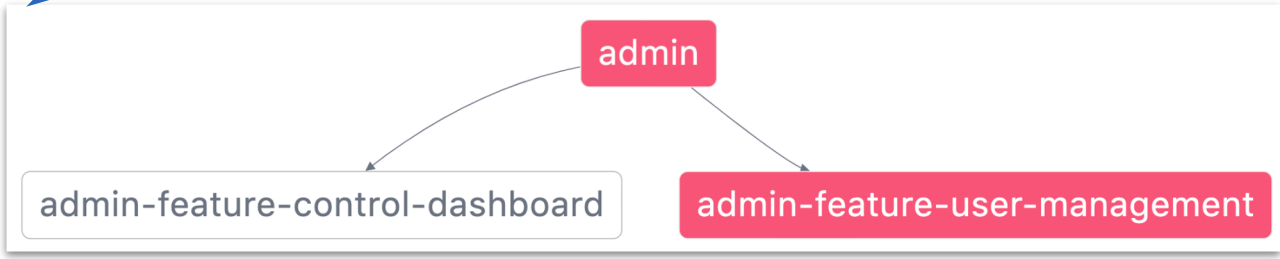



```
fundamentalconf git:(main) ✗ nx run admin-feature-control-dashboard:test
```

- Nx's main building block - libraries
- Each lib can run tasks separately
- All task runs are cached
- Project graph, affected
- Distributed Task Execution

```
● → fundamentalconf git:(main) ✗ nx test shared-util-format-number
  > nx run shared-util-format-number:test

  > NX  Successfully ran target test for project shared-util-format-number (5s)

    See Nx Cloud run details at https://nx.app/runs/88GZCiQ1na

● → fundamentalconf git:(main) ✗ nx test shared-util-format-number
  > nx run shared-util-format-number:test  [local cache]

  > NX  Successfully ran target test for project shared-util-format-number (56ms)

    Nx read the output from the cache instead of running the command for 1 out of 1 tasks.

    See Nx Cloud run details at https://nx.app/runs/RVmN7ws6qy
```





If you have any questions or just want to chat, drop me a message:

dmitry.stepanenko@valor-software.com

https://www.linkedin.com/in/stepanenko-dmitriy

https://medium.com/@stepanenkodmitri

https://github.com/dmitry-stepanenko

Code, that was used in this talk: https://github.com/dmitry-stepanenko/fundamentalconf