Turborepo

Monorepo task runner

START HERE

`turbo run build`
Build every package

`turbo build`
Shorthand for run

`turbo run test lint`
Several tasks at once

`turbo run build --filter=<pkg>`
One package only

`turbo run build --affected`
Only what changed

`turbo run build --dry`
Preview, run nothing

`turbo run build --force`
Ignore the cache

`turbo login`
Enable remote cache

turbo.json task graph → hash inputs → **Local cache** .turbo/cache → miss → **Remote cache** shared → miss → **Run task** store outputs

CONCEPTS

Turborepo hashes a task's inputs; an unchanged hash replays the cached outputs and logs instead of running.

Tasks live in turbo.json; package.json scripts of the same name are what actually runs.

Commands are run from anywhere in the repo; package-level turbo.json files extend the root with ["//"].

RUN TASKS

`turbo run <task>`
Run a task in all packages

`turbo run <task> <task>`
Run several tasks

`turbo run <pkg>#<task>`
Run one task in one package

`turbo run <task> --force`
Ignore cache, re-run everything

`turbo run <task> --dry`
Show what would run

`turbo run <task> --dry=json`
Dry run as JSON

`turbo run <task> --only`
Skip dependency tasks

`turbo run <task> --continue`
Keep going after a failure

`turbo run <task> --concurrency <n>`
Cap parallel tasks (number or %)

`turbo run <task> --summarize`
Write a run summary to .turbo/runs

`turbo run <task> --graph=<file>.html`
Write the task graph (dot/svg/html/mermaid)

`turbo run <task> --ui <stream|tui>`
Pick terminal UI

`turbo run <task> --output-logs <mode>`
Set log verbosity
Modes: full, hash-only, new-only, errors-only, none. Overrides outputLogs in turbo.json.

`turbo run <task> --env-mode loose`
Pass all env vars to tasks

RUN TASKS (cont.)

`turbo watch <task>`
Re-run tasks on file changes
Persistent tasks are ignored by watch; use the tool's own watcher for those.

FILTER

`--filter=<pkg>`
Select a package by name

`-F <pkg>`
Short form of --filter

`--filter="./apps/*"`
Select by directory glob

`--filter=<pkg>...`
Package and its dependencies

`--filter=...<pkg>`
Package and its dependents

`--filter=^<pkg>...`
Dependencies only, not the package

`--filter="[main]"`
Packages changed since a git ref

`--filter="...[HEAD^1]"`
Changed packages plus dependents

`--filter=!<pkg>`
Exclude a package

`--filter=<a> --filter=<b>`
Union of selections

`--affected`
Changed vs main, plus dependents
Compares main...HEAD. Override with TURBO_SCM_BASE and TURBO_SCM_HEAD. A shallow clone makes everything count as changed.

`turbo ls`
List packages in the repo

`turbo ls --affected`
List affected packages

CACHE

`turbo login`
Log in to the remote cache

`turbo link`
Link repo to a remote cache

`turbo unlink`
Disconnect from remote cache

`turbo logout`
Log out of the provider

`turbo link --scope <team>`
Link to a given team

CACHE (cont.)

`turbo login --sso-team <team>`
Log in to an SSO team

`turbo login --manual`
Enter a token by hand

`--cache=local:rw,remote:r`
Set cache read/write access

`--cache-dir <path>`
Change local cache location

`--token <token> --team <slug>`
Remote cache credentials for CI
Or set TURBO_TOKEN and TURBO_TEAM in the environment.

REPO TOOLS

`turbo prune <pkg>`
Create a partial monorepo in out/

`turbo prune <pkg> --docker`
Docker-layer friendly output
Creates json/ with package.json files and full/ with sources, plus the pruned lockfile.

`turbo prune <pkg> --out-dir <path>`
Change prune output directory

`turbo prune <pkg> --production`
Skip devDependency-only packages

`turbo generate`
Run a custom generator

`turbo generate workspace`
Scaffold a new package or app

`turbo gen workspace --copy <name>`
Copy an existing workspace

`turbo gen workspace --name <n> --type <app|package>`
Scaffold non-interactively

`turbo boundaries`
Check package boundary rules

`turbo query`
Run a GraphQL query on the repo

`npx create-turbo@latest`
Start a new monorepo

TURBO.JSON

`"tasks": { ... }`
Define runnable tasks

`"dependsOn": ["^build"]`
Run dependencies' build first

TURBO.JSON (cont.)

`"dependsOn": ["lint"]`
Run same-package task first

`"outputs": ["dist/**"]`
Files to cache after success
Without outputs only the logs are cached.

`"inputs": ["src/**"]`
Files that affect the hash

`"cache": false`
Never cache this task

`"persistent": true`
Long-running task such as dev

`"env": ["API_URL"]`
Env vars hashed into the task

`"globalEnv": ["CI"]`
Env vars hashed into every task

`"globalDependencies": [".env"]`
Files that invalidate all caches

`"passThroughEnv": ["<VAR>"]`
Pass env vars without hashing

`"extends": ["//"]`
Inherit root config in a package

Dependency syntax

Entry	Runs first
`build`	build in same package
`^build`	build in dependencies
`app#build`	build in package app

RECIPES

Typical turbo.json

```
{
  "tasks": {
    "build": { "dependsOn":
["^build"], "outputs": ["dist/
**"] },
    "test": { "dependsOn":
["build"] },
    "dev": { "cache": false,
"persistent": true }
  }
}
```

CI: only what a PR touched

```
turbo run build test lint --
affected
# shallow clones mark everything
changed:
# use fetch-depth: 0 or a
blobless clone
```

Remote cache in CI

```
export TURBO_TOKEN=<token>
export TURBO_TEAM=<team>
turbo run build
```

Docker build of one app

```
turbo prune web --docker
# copy out/json then install,
then out/full
# then: turbo run build --
filter=web
```