Ninja Fast build executor

CMakeLists / meson.build you write → `cmake / meson setup` → **build.ninja** generated → `ninja` → **Outputs** only what is stale

CONCEPTS

> Ninja is a fast executor, not a build language: a generator such as CMake or Meson writes build.ninja for you.

> Ninja skips the work when outputs are newer than their inputs and the recorded command is unchanged.

> Do not hand-edit generated build.ninja; change the generator input and it regenerates itself.

BUILDING

`ninja`
Build the default targets

`ninja <target>`
Build only the named target(s)

`ninja -C <dir>`
Change to a directory first

`ninja -j <N>`
Run N jobs in parallel

`ninja -n`
Dry run, do not execute

`ninja -v`
Show full command lines

BUILDING (cont.)

`ninja -k <N>`
Keep going until N jobs fail

`ninja -f <file>`
Use another build file

`ninja -l <N>`
Hold back at load average above N

`ninja --quiet`
Hide progress, show command output

`ninja --version`
Print the Ninja version

`ninja -h`
Show help and options

INSPECTING THE GRAPH

`ninja -t targets all`
List every target and its rule

`ninja -t targets rule <rule>`
List targets built by a rule

`ninja -t targets depth <n>`
List targets down to a depth

`ninja -t commands <target>`
Print commands to rebuild a target

INSPECTING THE GRAPH (cont.)

`ninja -t query <path>`
Show a file's inputs and outputs

`ninja -t inputs <target>`
List all inputs of a target

`ninja -t graph <target>`
Emit a Graphviz dot file

`ninja -t rules`
List all rules

`ninja -t deps <target>`
Show deps recorded from depfiles

`ninja -t missingdeps`
Find missing generated-file deps

CLEANING AND TOOLING

`ninja -t clean`
Remove all built files

`ninja -t clean <target>`
Clean one target and its deps

`ninja -t clean -r <rule>`
Clean outputs of a rule

`ninja -t compdb <rule> ...`
Dump compile_commands.json

DEBUGGING AND ENV

`ninja -d explain`
Say why each command reruns

`ninja -d stats`
Print operation counts and timing

`NINJA_STATUS="[%f/%t %e] " ninja`
Customize progress line

`ninja -j1 -v`
Serial verbose build to find a failure

WRITING BUILD.NINJA

`rule <name>`
Declare a rule with a command

`command = cc -c $in -o $out`
Rule command using $in and $out

`build <out>: <rule> <in>`
Declare a build edge

`build <out>: <rule> <in> | <dep>`
Add an implicit dependency

`build <out>: <rule> <in> || <dep>`
Add an order-only dependency

WRITING BUILD.NINJA (cont.)

`default <target>`
Set what a bare ninja builds

`build <name>: phony <targets>`
Alias for other targets

`depfile = $out.d`
Read header deps from a depfile

`description = CC $out`
Short text shown while building

`subninja <file>`
Include a file in a new scope

RECIPES

Minimal hand-written build

```
rule cc
  command = cc -c $in -o $out
  depfile = $out.d
  deps = gcc
  description = CC $out
rule link
  command = cc $in -o $out
build main.o: cc main.c
build app: link main.o
default app
```

Meson or CMake out-of-tree

```
meson setup build && ninja -C build
# or
cmake -G Ninja -S . -B build
ninja -C build
```