# Ninja Fast build executor

**START HERE**

**`cmake -G Ninja -B build`**
Generate a build.ninja

**`ninja -k 0`**
Keep going past failures

**`ninja -C build`**
Build the default targets

**`ninja -j <N>`**
Set parallel jobs

**`ninja -C build <target>`**
Build one target

**`ninja -t targets all`**
List what can be built

**`ninja -n -v`**
Preview the commands

**`ninja -t clean`**
Remove built files

**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
*Typical out-of-source use: ninja -C build.*

**`ninja -j <N>`**
Run N jobs in parallel
*0 means unlimited; the default follows the CPU count.*

**`ninja -n`**
Dry run, do not execute

**`ninja -v`**
Show full command lines

**`ninja -k <N>`**
Keep going until N jobs fail
*0 means never stop; default is 1.*

**`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

## INSPECTING THE GRAPH (cont.)

**`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

**`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

**`ninja -t browse <target>`**
Browse the graph in a browser
*Needs Python; --port and --no-browser are available.*

**`ninja -t multi-inputs <target>`**
Map targets to their inputs

**`ninja -t list`**
List all subtools

## 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 clean -g`**
Also remove generated files

**`ninja -t cleandead`**
Remove outputs no longer in manifest

**`ninja -t compdb <rule> ...`**
Dump compile_commands.json
*Redirect to a file: ninja -t compdb cc > compile_commands.json (rule names depend on the generator).*

## CLEANING AND TOOLING (cont.)

**`ninja -t compdb-targets <target>`**
Compdb limited to given targets

**`ninja -t recompact`**
Compact the deps and build logs

**`ninja -t restat`**
Re-stat outputs in the build log

## DEBUGGING AND ENV

**`ninja -d explain`**
Say why each command reruns

**`ninja -d stats`**
Print operation counts and timing

**`ninja -d keepdepfile`**
Keep depfiles after reading

**`ninja -d keeprsp`**
Keep response files on success

**`ninja -d list`**
List debug modes

**`ninja -w list`**
List warning flags

**`NINJA_STATUS="[%f/%t %e] " ninja`**
Customize progress line
*%f finished, %t total, %p percent, %e elapsed seconds.*

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

## WRITING BUILD.NINJA

**`rule <name>`**
Declare a rule with a command
*Indented lines set command, description, depfile, deps.*

**`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

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

## WRITING BUILD.NINJA (cont.)

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

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

**`deps = gcc`**
Store depfile deps in the deps log

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

**`pool <name>`**
Declare a pool, set depth to cap jobs

**`restat = 1`**
Skip dependents if output is unchanged

**`generator = 1`**
Mark a rule as the regenerator

**`include <file>`**
Include a file in the same scope

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

**`ninja_required_version = 1.10`**
Require a minimum Ninja

**`builddir = <dir>`**
Directory for .ninja_log and deps log

## 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
```

### Compilation database for clangd

```
ninja -C build -t compdb cc cxx
> compile_commands.json
```

### Find why it rebuilds

```
ninja -d explain -n
ninja -t query <path>
```