watchexec Run on file change

START HERE

`watchexec <cmd>`
Run on any change

`watchexec -c <cmd>`
Clear screen first

`watchexec -e py pytest`
Only Python files

`watchexec -p <cmd>`
Wait for first change

`watchexec -r -- <server>`
Restart on change

`watchexec -i "*.log" <cmd>`
Ignore files

`watchexec -w src <cmd>`
Watch one folder

`watchexec -n <cmd>`
Run without a shell

File event save, create, delete → `debounce` → **Filters** paths, ignores, -e, -f → `pass` → **Command** start or restart → `exit` → **Wait** for next change

CONCEPTS

Put -- before your command so its own flags are not read by watchexec.

By default it watches the current directory recursively and honours .gitignore and .ignore, so build output folders are usually skipped already.

The default shell is $SHELL (sh if unset); with -n the command is split on spaces, so no pipes, globs or redirects.

RUNNING

`watchexec <cmd> [args]`
Run a command whenever files change

`watchexec -- <cmd> --flag`
Stop option parsing before the command

`watchexec -p <cmd>`
Postpone: skip the initial run

`watchexec -r <cmd>`
Restart the command on change

`watchexec -c <cmd>`
Clear the screen before each run

`watchexec -n <cmd>`
No shell: run the program directly

`watchexec --shell <shell> <cmd>`
Choose the shell, e.g. bash or zsh

`watchexec -E <KEY=VALUE> <cmd>`
Set an env var for the command only

`watchexec --workdir <dir> <cmd>`
Run the command in another directory

`watchexec -I <cmd>`
Interactive: r restart, s stop, p pause, q quit

`watchexec --exit-on-error <cmd>`
Quit when the command fails

`watchexec --timeout <time> <cmd>`
Stop a run that takes too long

RUNNING (cont.)

`watchexec --delay-run <time> <cmd>`
Sleep before each run

`watchexec --timings <cmd>`
Print how long each run took

`watchexec -q <cmd>`
Hide watchexec's own start/stop messages

`watchexec -N <cmd>`
Desktop notification when the run starts or ends

`watchexec --bell <cmd>`
Ring the terminal bell when done

WHAT TO WATCH

`-w <path>`
Watch a file or directory (repeatable)

`-W <dir>`
Watch a directory without recursion

`-e <ext,ext>`
Only react to these extensions
Give them without dots: -e rs,toml

`-f <glob>`
Only react to paths matching a pattern

`-i <glob>`
Ignore paths matching a pattern

`--ignore-file <path>`
Load ignore patterns from a file

`--filter-file <path>`
Load filter patterns from a file

`-F <file>`
Read watch paths from a file (- for stdin)

`--no-vcs-ignore`
Watch files that .gitignore excludes

`--no-default-ignore`
Also watch .git, swap files, .DS_Store

`--no-discover-ignore`
Skip loading every ignore file

`--ignore-nothing`
Ignore no files, not even defaults

`--project-origin <dir>`
Override the detected project root

WHAT TO WATCH (cont.)

`--fs-events <list>`
Filter by event: create, remove, rename, modify, access

`--no-meta`
Skip metadata-only events

`-j <expr>`
Filter events with a jaq program

BUSY AND SIGNALS

`-o <mode>`
If busy: do-nothing, queue, restart or signal

`-r`
Same as -o restart

`-s <signal>`
Send a signal to the running command on change

`--stop-signal <signal>`
Signal used to stop it (default SIGTERM)

`--stop-timeout <time>`
Grace time before a force kill (default 10s)

`--map-signal <in:out>`
Translate a received signal to another

`--wrap-process <mode>`
auto, group, session or none

`--stdin-quit`
Exit when stdin closes

DEBOUNCE AND POLLING

`-d <time>`
Wait for quiet before acting (default 50ms)

`--poll [interval]`
Poll instead of OS events (network drives)

`--no-follow-symlinks`
Do not follow directory symlinks

CHANGED PATHS

`--emit-events-to <mode>`
none, environment, stdio, file, json-stdio, json-file

`--emit-events-to=json-stdio`
Send events as JSON lines on stdin

CHANGED PATHS (cont.)

`--emit-events-to=environment`
Set WATCHEXEC_*_PATH variables (deprecated)
Variables: WATCHEXEC_COMMON_PATH plus _CREATED_, _REMOVED_, _RENAMED_, _WRITTEN_, _META_CHANGED_ and _OTHERWISE_CHANGED_PATH

`--print-events`
Print the events that triggered a run

`--only-emit-events`
Emit events only; run no command

SETUP AND DEBUG

`watchexec --completions <shell>`
Print shell completions

`watchexec --manual`
Show the full manual

`watchexec -v / -vvvv`
More diagnostic logging

`WATCHEXEC_LOG=<level>`
Set log level (beats -v)

`--log-file [path]`
Write JSON logs to a file

`--color <mode>`
auto, always or never (NO_COLOR works too)

`watchexec @<argfile>`
Load arguments from a file, one per line

`watchexec -V`
Show the version

RECIPES

Run tests on Rust changes

```
watchexec -c -e rs,toml -- cargo test
```

Dev server with restart

```
watchexec -r -e py,html -- python manage.py runserver --noreload
```

Rebuild assets, skip output

```
watchexec -w src -i "dist/**" -e js,css -- npm run build
```

Debounced build, then notify

```
watchexec -d 500 -N -p -e md -- make site
```