air Go live reload

START HERE

`go install github.com/air-verse/air@latest`
Install air

`air -d`
Print all logs

`air init`
Write a default .air.toml

`air -v`
Show the version

`air`
Watch, rebuild, restart

`air -- <args>`
Pass args to your binary

`air -c <file>`
Use another config file

`air --build.cmd "<cmd>"`
Override a config key

File change watched ext/dir → **`delay`** → **build.cmd** go build -o ./tmp/main . → **`stop old, start`** → **entrypoint** your app → **`proxy`** → **Browser** reloads

CONCEPTS

air watches your files, runs the build command on each change, then restarts the built binary. It reads .air.toml in the current directory and uses defaults without one.

Air's own flags come first; everything after -- goes to your program.

INSTALL

`go install github.com/air-verse/air@latest`
Install into GOPATH/bin

`export PATH=$PATH:$(go env GOPATH)/bin`
Fix 'command not found: air'

`go get -tool github.com/air-verse/air@latest`
Add as a project-local tool
Run it afterwards with go tool air.

`go tool air -v`
Run the project-local air

`brew install go-air`
Install with Homebrew

`mise use -g air`
Install with mise

RUN

`air init`
Create .air.toml with defaults

`air`
Run with .air.toml or defaults

`air -c <file>`
Use a specific config file

`air -d`
Debug mode, print all logs

`air -v`
Print the version

`air -h`
List every flag

`air server --port 8080`
Run the binary with these args
Runs ./tmp/main server --port 8080.

`air -c <file> -- -h`
Split air's flags from the app's

FLAG OVERRIDES

`air --build.cmd "go build -o ./tmp/main ./cmd/api"`
Set the build command

`air --build.entrypoint ./tmp/main`
Set the binary to run

`air --build.exclude_dir node_modules,vendor`
Exclude dirs (comma list)
List flags take commas or can be repeated; values append.

FLAG OVERRIDES (cont.)

`air --build.include_ext go,html`
Watch these extensions

`air --build.delay 500`
Debounce rebuilds (ms)

`air --proxy.enabled true --proxy.app_port 8080`
Turn on the browser proxy

[BUILD] SETTINGS

`cmd = "go build -o ./tmp/main ."`
Command that builds the app

`entrypoint = ["./tmp/main"]`
Executable plus default args

`args_bin = ["-v"]`
Extra args for the binary

`bin = "tmp/main"`
Old way to name the binary
Deprecated; use entrypoint.

`full_bin = "APP_ENV=dev ./tmp/main"`
Run with env vars set

`include_ext = ["go", "tpl", "html"]`
Extensions to watch
[""] watches all.*

`exclude_dir = ["tmp", "vendor"]`
Directories to ignore

`include_dir = ["cmd", "internal"]`
Only watch these dirs

`include_file = ["go.mod"]`
Also watch specific files

`exclude_file = ["gen.go"]`
Ignore specific files

`exclude_regex = ["_test\.go"]`
Ignore paths by regex

`exclude_unchanged = true`
Skip files with same content

`follow_symlink = true`
Follow symlinked dirs

`delay = 1000`
Debounce before rebuild (ms)

`stop_on_error = true`
Stop old binary if build fails

`send_interrupt = true`
Interrupt before killing the app
Ignored on Windows.

`kill_delay = 500`
Wait after the interrupt
The example file says nanoseconds; likely milliseconds.

[BUILD] SETTINGS (cont.)

`poll = true`
Poll instead of fsnotify
Useful on network or container mounts.

`poll_interval = 500`
Poll interval in ms
Minimum is 500.

`pre_cmd = ["go generate ./..."]`
Commands before each build

`post_cmd = ["echo bye"]`
Commands after Ctrl+C

TOP-LEVEL & MISC

`root = "."`
Working directory to watch

`tmp_dir = "tmp"`
Folder for build output and logs

`env_files = [".env"]`
Load .env files; later wins
Existing environment variables are not overwritten.

`[misc] clean_on_exit = true`
Delete tmp dir on exit

`[log] time = true`
Timestamps in air's logs

`[log] main_only = true`
Show only app output

`[log] silent = true`
Silence air's own logs

`[screen] clear_on_rebuild = true`
Clear the terminal per rebuild

`[build.linux] cmd = "..."`
Per-OS override block
Also build.darwin and build.windows; only cmd, pre_cmd, post_cmd, bin, entrypoint, full_bin, args_bin are allowed.

On Windows, escape backslashes in TOML strings: "tmp\\main.exe".

BROWSER RELOAD & RULES

`[proxy] enabled = true`
Reload the browser after rebuilds

`proxy_port = 8090`
Port you open in the browser

`app_port = 8080`
Port your app listens on

`app_start_timeout = 5000`
Wait for app after rebuild (ms)
Raise it on 'unable to reach app' errors.

BROWSER RELOAD & RULES (cont.)

The proxy injects its script before </body>; HTML without it is served unchanged. Static assets must match include_dir, include_ext or include_file.

`[[build.rules]]`
Run a command instead of a rebuild
Needs at least one include_* matcher. Fields: name, cmd, delay (default 1000).

`include_dir = ["web"]  # inside a rule`
Rule dirs watched even if excluded

`cmd = "/usr/bin/true"`
Build only, never run the app

DOCKER

`docker run -it --rm -w /app -v $(pwd):/app -p 8080:8080 cosmtrek/air`
Run the official image

`... cosmtrek/air -c .air.toml`
Pass a config to the image

`CMD ["air", "-c", ".air.toml"]`
Dockerfile dev command

In Compose, set working_dir to match the mounted volume path.

RECIPES

Web app with browser reload

```
air init
# edit .air.toml:
#   [build] include_ext = ["go", "html", "css"]
#   [proxy] enabled = true, proxy_port = 8090, app_port = 8080
air
# open http://localhost:8090
```

Several mains, one config

```
air --build.cmd "go build -o ./tmp/api ./cmd/api" \
    --build.entrypoint ./tmp/api
```

Rebuild frontend without restarting Go

```
[[build.rules]]
name = "assets"
cmd = "npm run build"
include_dir = ["web"]
include_ext = ["js", "ts", "css"]
delay = 1000
```

Containers and network mounts

```
air --build.poll true --build.poll_interval 500 \
    --build.exclude_dir node_modules,vendor,tmp
```