GoReleaser

Ship Go releases

START HERE

`goreleaser init`
Create .goreleaser.yaml

`goreleaser check`
Validate the config

`goreleaser build --snapshot --clean`
Build only, no tag needed

`goreleaser release --snapshot --clean`
Dry run of a full release

`git tag -a v0.1.0 -m "<msg>"`
Tag the version

`git push origin v0.1.0`
Push the tag

`export GITHUB_TOKEN=<token>`
Allow publishing

`goreleaser release --clean`
Build and publish

Git tag v1.2.3 → `build` → **Binaries** per OS/arch → `archive` → **Archives** + checksums → `sign` → **Signatures** optional → `publish` → **GitHub release** + changelog

What a release does

CONCEPTS

GoReleaser releases the latest git tag. Use semver tags (v1.2.3) and a clean working tree, or add --snapshot to skip the checks.

Everything lands in dist/. Use --clean on every run so old artifacts do not mix with new ones.

SETUP

`goreleaser init`
Write an example .goreleaser.yaml

`goreleaser check`
Validate the config file

`goreleaser healthcheck`
Check that needed tools are installed

`goreleaser --help`
List commands and global flags

`goreleaser <command> --help`
Flags of one command

`goreleaser --version`
Show the installed version

`goreleaser --config <file> <command>`
Use a custom config path

BUILD

`goreleaser build --snapshot --clean`
Build all targets without a tag

`goreleaser build --single-target --clean`
Build only for this machine

`GOOS=<os> GOARCH=<arch> goreleaser build --single-target`
Build one chosen target

`goreleaser build --id <build>`
Build one build ID

`goreleaser build --output <path>`
Copy the binary to a path

RELEASE

`goreleaser release --clean`
Publish the release for the current tag

`goreleaser release --snapshot --clean`
Full local run, nothing published

`goreleaser release --skip=publish`
Build and package but do not publish

`goreleaser release --skip=<step>,<step>`
Skip several steps

RELEASE (cont.)

`goreleaser release --draft`
Create the release as a draft

`goreleaser release --release-notes <file>`
Use your own release notes

`goreleaser release --auto-snapshot`
Snapshot if the tree is dirty

`git tag -a v<x.y.z> -m "<msg>"`
Create the release tag

`git push origin v<x.y.z>`
Push the tag

`GITHUB_TOKEN=<token> goreleaser release`
Pass the token for one run

Classic token needs repo scope; fine-grained needs contents: write

CONFIG KEYS

`version: 2`
First line of the config file

`project_name: <name>`
Name used in templates and archives

`builds:`
Binaries to compile

`archives:`
How binaries are packaged

`checksum:`
Checksum file settings

CONFIG KEYS (cont.)

`changelog:`
Release notes from commits

`release:`
GitHub/GitLab/Gitea release options

`signs:`
Sign artifacts with an external tool

`before: hooks:`
Commands run before the build

BUILDS

`main: ./cmd/<name>`
Path of the main package

`binary: <name>`
Output binary name

`goos: [linux, darwin, windows]`
Target operating systems

`goarch: [amd64, arm64]`
Target architectures

`env: [CGO_ENABLED=0]`
Build environment variables

`ldflags: [-s -w -X main.version={{.Version}}]`
Linker flags with version stamp

`flags: [-trimpath]`
Extra go build flags

`ignore: [{goos: <os>, goarch: <arch>}]`
Skip one OS/arch pair

BUILDS (cont.)

`id: <build>`
Name a build to reference it elsewhere

ARCHIVES AND CHECKSUMS

`formats: [tar.gz]`
Archive formats (tar.gz is default)

`name_template: "{{ .ProjectName }}_{{ .Version }}_{{ .Os }}_{{ .Arch }}"`
Archive file name

`format_overrides: [{goos: windows, formats: [zip]}]`
Zip for Windows only

`files: [LICENSE*, README*]`
Extra files inside the archive

`checksum: name_template: checksums.txt`
Name of the checksum file

CHANGELOG AND RELEASE

`changelog: sort: asc`
Order commits in the notes

`changelog: filters: exclude: ["^docs:"]`
Hide commits matching a regex

CHANGELOG AND RELEASE (cont.)

`changelog: use: github-native`
Let GitHub generate the notes

Turns off groups, sort and filters

`release: draft: true`
Leave the release as a draft

`release: prerelease: auto`
Mark rc/beta tags as pre-releases

TEMPLATES AND SIGNING

`{{ .Version }}`
Version without the leading v

`{{ .Tag }}`
Git tag as pushed

`{{ .ProjectName }}`
Project name

RECIPES

Minimal .goreleaser.yaml

```
version: 2
builds:
  - env: [CGO_ENABLED=0]
    goos: [linux, darwin, windows]
    goarch: [amd64, arm64]
    ldflags: [-s -w -X main.version={{.Version}}]
archives:
  - formats: [tar.gz]
    format_overrides:
      - {goos: windows, formats: [zip]}
```