Go Go toolchain commands

CONCEPTS

A module (go.mod) is a tree of packages; a package is one directory of .go files. Commands take package patterns: . is the current package, ./... is it plus everything below.

Module go.mod, versioned unit
Package one directory, one import path
.go file declarations; _test.go for tests

Module, package and file

go build only checks and discards non-main packages; for a main package it writes a binary. go install puts the binary in $GOBIN (default $GOPATH/bin).

go.sum holds checksums of dependencies: commit it together with go.mod.

BUILD AND RUN

go run .
Compile and run the package here

go run *<file.go> <args>*
Run files, passing program args

go build ./...
Compile all packages, discard results

go build -o *<path>* **.**
Write the binary to a chosen path

go install *<pkg>*@latest
Install a tool binary into $GOBIN

go build -race
Build with the race detector

go build -ldflags="-s -w"
Strip symbols and debug info

go build -ldflags="-X main.version=*<v>***"**
Set a string variable at link time

go build -tags *<tag,...>*
Enable build tags

go build -trimpath
Remove local paths from the binary

go clean -cache
Delete the build cache

BUILD AND RUN (cont.)

go clean -testcache
Expire cached test results

CROSS-COMPILING

GOOS=*<os>* **GOARCH=***<arch>* **go build**
Build for another platform

go tool dist list
List supported GOOS/GOARCH pairs

CGO_ENABLED=0 go build
Static build without C toolchain

TESTING

go test
Test the package in the current dir

go test ./...
Test every package below here

go test -v
Print each test and its output

go test -run *<regexp>*
Run only matching tests

go test -count=1
Bypass the test result cache

go test -race
Run with the race detector

go test -short
Skip tests that check testing.Short()

go test -failfast
Stop after the first failure

go test -timeout *<d>*
Abort a test binary after d

go test -skip *<regexp>*
Skip matching tests

go test -shuffle=on
Randomise test order

go test -cover
Show statement coverage

go test -coverprofile=*<file>*
Write a coverage profile

go tool cover -html=*<file>*
Browse coverage in a browser

go test -bench *<regexp>*
Run benchmarks (. for all)

go test -benchmem
Add allocation stats to benchmarks

FORMAT AND CHECK

go fmt ./...
Format files with gofmt -l -w

go vet ./...
Find likely mistakes

go doc *<pkg>*[.*<Symbol>*]
Show docs for a package or symbol

go generate ./...
Run //go:generate directives

DEPENDENCIES

go get *<pkg>*
Add a dependency at the latest version

go get *<pkg>*@*<version>*
Pin a version, branch or commit

go get *<pkg>*@none
Remove a dependency

go get -u ./...
Upgrade deps of this module

go list -m all
List all modules in the build

go list -m -u all
Show modules with available updates

MODULES

go mod init *<module>*
Create go.mod here

go mod tidy
Add missing, remove unused modules

go mod edit -replace *<old>*=*<new>*
Add a replace directive

WORKSPACES

go work init *<dir>*...
Create go.work for local modules

go work use *<dir>*
Add a module to the workspace

ENVIRONMENT

go version
Print the Go toolchain version

go env
Print all Go environment settings

go env -w *<VAR>*=*<value>*
Persist a setting