m... 37 Branches 151 Tags Go to file Go to file Code

DePuina and francislavoie logging: add DirMode options and propagate FileMode to rotations (#...

294dfff · 7 hours ago 2,475 Commits

.github	chore: bump Go to v1.26 (#7466)	2 weeks ago
caddyconfig	httpcaddyfile: Fix missing TLS conne...	2 days ago
caddytest	logging: add DirMode options and pr...	7 hours ago
cmd	chore: Add nolints to work around h...	last week
internal	logging: Adjustments to BufferedLog...	5 months ago
modules	logging: add DirMode options and pr...	7 hours ago
notify	notify: implement windows service st...	2 months ago
.editorconfig	caddytest: Rename adapt tests to *....	2 years ago
.gitattributes	chore: Add .gitattributes to force *.g...	4 years ago
.gitignore	gitignore: Add rule for caddyfile.go (...	2 years ago
.golangci.yml	ci: set proper build tags in golangci a...	6 months ago
.goreleaser.yml	chore: Disable windows/arm build ta...	3 days ago
.pre-commit-config.yaml	chore: apply security best practices f...	8 months ago
AUTHORS	Add authors file	7 years ago
LICENSE	Add license	7 years ago
README.md	Update detail in readme	2 weeks ago
admin.go	admin: Fix tests locally, properly isola...	last week
admin_test.go	admin: Fix tests locally, properly isola...	last week
caddy.go	chore: fix some comments to improv...	last week
caddy_test.go	core: Check for nil event origin (#7047)	9 months ago
context.go	core: custom slog handlers for modu...	3 months ago
context_test.go	chore: Bump up to Go 1.19, minimu...	4 years ago
duration_fuzz.go	core: Windows service integration (#...	4 years ago
filepath.go	chore: make FastAbs comment more...	2 years ago
filepath_windows.go	chore: make FastAbs comment more...	2 years ago
filesystem.go	core: Implement FastAbs to avoid re...	2 years ago
go.mod	go.mod: Upgrade dependencies	3 days ago
go.sum	go.mod: Upgrade dependencies	3 days ago
listen.go	chore: ugh, lint fix... (#7275)	5 months ago
listen_unix.go	core: Implement socket activation lis...	2 years ago
listen_unix_setopt.go	Switch Solaris-derivatives away from ...	3 years ago
listen_unix_setopt_freebsd.go	core: Use SO_REUSEPORT_LB on Free...	3 years ago
listeners.go	Revert "listeners: Add support for na...	4 days ago
listeners_fuzz.go	core: Windows service integration (#...	4 years ago
listeners_test.go	Revert "listeners: Add support for na...	4 days ago
logging.go	logging: Adjustments to BufferedLog...	5 months ago
logging_test.go	log: default logger should respect {i...	10 months ago
metrics.go	metrics: scope metrics to active confi...	2 years ago
modules.go	core: Show JSON error offsets where ...	last month

About

Fast and extensible multi-platform HTTP/1-2-3 web server with automatic HTTPS

caddyserver.com

#go #tls #golang #http #security #privacy #web-server #https #acme #reverse-proxy #http-server #caddy #http3 #caddyfile #automatic-https

Readme

Apache-2.0 license

Contributing

Security policy

Activity

Custom properties

70.3k stars

839 watching

4.6k forks

Report repository

Releases 133

v2.11.1 Latest
10 hours ago

+ 132 releases

Sponsor this project

mholt Matt Holt

Sponsor

Learn more about GitHub Sponsors

Used by 1.8k

+ 1,786

Contributors 373

+ 359 contributors

Languages

Go 97.8% HTML 2.2%

modules_test.go	Couple of minor fixes, update readme	7 years ago
replacer.go	chore: fix struct name in comment (#...	7 months ago
replacer_fuzz.go	core: Windows service integration (#...	4 years ago
replacer_test.go	use a more modern writing style to s...	6 months ago
service_windows.go	ci: use gci linter (#5708)	3 years ago
sigtrap.go	go.mod: Upgrade CertMagic to v0.16.0	4 years ago
sigtrap_nonposix.go	core: Windows service integration (#...	4 years ago
sigtrap_posix.go	core: Reloading with `SIGUSR1` if confi...	5 months ago
storage.go	pki: Add trust subcommand to install...	6 years ago
usagepool.go	chore: ugh, lint fix... (#7275)	5 months ago

README Contributing Apache-2.0 license Security

a ZeroSSL project

Every site on HTTPS

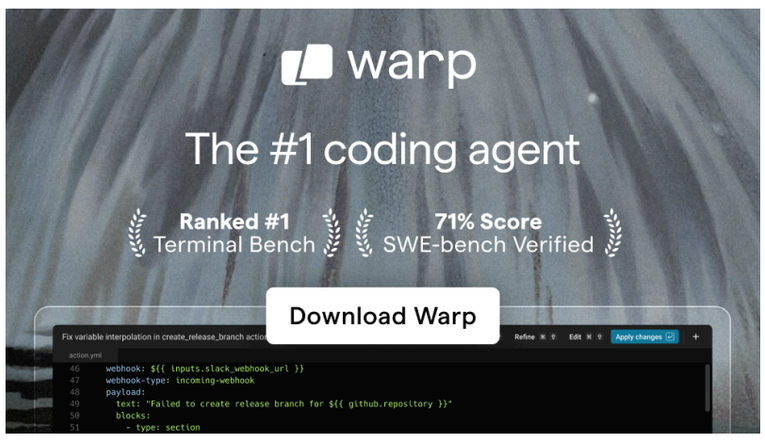
Caddy is an extensible server platform that uses TLS by default.

Releases · Documentation · Get Help

Tests passing | openssf best practices in progress 94% | godoc reference | Follow @caddyserver | community forum | used by 161 projects | OSS hosting by cloudsmith

Powered by

Special thanks to:



Warp, built for coding with multiple AI agents

Available for MacOS, Linux, & Windows

Menu

Features

- **Easy configuration** with the Caddyfile
- **Powerful configuration** with its native JSON config
- **Dynamic configuration** with the JSON API
- **Config adapters** if you don't like JSON
- **Automatic HTTPS** by default
  - ZeroSSL and Let's Encrypt for public names
  - Fully-managed local CA for internal names & IPs
  - Can coordinate with other Caddy instances in a cluster
  - Multi-issuer fallback
  - Encrypted ClientHello (ECH) support
- **Stays up when other servers go down** due to TLS/OCSP/certificate-related issues
- **Production-ready** after serving trillions of requests and managing millions of TLS certificates
- **Scales to hundreds of thousands of sites** as proven in production
- **HTTP/1.1, HTTP/2, and HTTP/3** all supported by default
- **Highly extensible** modular architecture lets Caddy do anything without bloat
- **Runs anywhere** with **no external dependencies** (not even libc)
- Written in Go, a language with higher **memory safety guarantees** than other servers
- Actually **fun to use**
- So much more to discover

Install

The simplest, cross-platform way to get started is to download Caddy from GitHub Releases and place the executable file in your PATH.

See our online documentation for other install instructions.

Build from source

Requirements:

- Go 1.25.0 or newer

For development

Note: *These steps will not embed proper version information. For that, please follow the instructions in the next section.*

```
$ git clone "https://github.com/caddyserver/caddy.git"
$ cd caddy/cmd/caddy/
$ go build
```

When you run Caddy, it may try to bind to low ports unless otherwise specified in your config. If your OS requires elevated privileges for this, you will need to give your new binary permission to do so. On Linux, this can be done easily with: `sudo setcap cap_net_bind_service=+ep ./caddy`

If you prefer to use `go run` which only creates temporary binaries, you can still do this with the included `setcap.sh` like so:

```
$ go run -exec ./setcap.sh main.go
```

If you don't want to type your password for `setcap`, use `sudo visudo` to edit your sudoers file and allow your user account to run that command without a password, for example:

```
username ALL=(ALL:ALL) NOPASSWD: /usr/sbin/setcap
```

replacing `username` with your actual username. Please be careful and only do this if you know what you are doing! We are only qualified to document how to use Caddy, not Go tooling or your computer, and we are providing these instructions for convenience only; please learn how to use your own computer at your own risk and make any needful adjustments.

Then you can run the tests in all modules or a specific one:

```
$ go test ./...
$ go test ./modules/caddyhttp/tracing/
```

With version information and/or plugins

Using our builder tool, `xcaddy` ...

```
$ xcaddy build
```

...the following steps are automated:

1. Create a new folder: `mkdir caddy`
2. Change into it: `cd caddy`
3. Copy Caddy's main.go into the empty folder. Add imports for any custom plugins you want to add.
4. Initialize a Go module: `go mod init caddy`
5. (Optional) Pin Caddy version: `go get github.com/caddyserver/caddy/v2@version` replacing `version` with a git tag, commit, or branch name.
6. (Optional) Add plugins by adding their import: `_ "import/path/here"`
7. Compile: `go build -tags=nobadger,nomysql,nopgx`

Quick start

The Caddy website has documentation that includes tutorials, quick-start guides, reference, and more.

We recommend that all users -- regardless of experience level -- do our Getting Started guide to become familiar with using Caddy.

If you've only got a minute, the website has several quick-start tutorials to choose from! However, after finishing a quick-start tutorial, please read more documentation to understand how the software works. 🙂

Overview

Caddy is most often used as an HTTPS server, but it is suitable for any long-running Go program. First and foremost, it is a platform to run Go applications. Caddy "apps" are just Go programs that are implemented as Caddy modules. Two apps -- `tls` and `http` -- ship standard with Caddy.

Caddy apps instantly benefit from automated documentation, graceful on-line config changes via API, and unification with other Caddy apps.

Although JSON is Caddy's native config language, Caddy can accept input from config adapters which can essentially convert any config format of your choice into JSON: Caddyfile, JSON 5, YAML, TOML, NGINX config, and more.

The primary way to configure Caddy is through its API, but if you prefer config files, the command-line interface supports those too.

Caddy exposes an unprecedented level of control compared to any web server in existence. In Caddy, you are usually setting the actual values of the initialized types in memory that power everything from your HTTP handlers and TLS handshakes to your storage medium. Caddy is also ridiculously extensible, with a powerful plugin system that makes vast improvements over other web servers.

To wield the power of this design, you need to know how the config document is structured. Please see our documentation site for details about Caddy's config structure.

Nearly all of Caddy's configuration is contained in a single config document, rather than being scattered across CLI flags and env variables and a configuration file as with other web servers. This makes managing your server config more straightforward and reduces hidden variables/factors.

Full documentation

Our website has complete documentation:

https://caddyserver.com/docs/

The docs are also open source. You can contribute to them here: https://github.com/caddyserver/website

Getting help

- We advise companies using Caddy to secure a support contract through Ardan Labs before help is needed.
- A sponsorship goes a long way! We can offer private help to sponsors. If Caddy is benefitting your company, please consider a sponsorship. This not only helps fund full-time work to ensure the longevity of the project, it provides your company the resources, support, and discounts you need; along with being a great look for your company to your customers and potential customers!
- Individuals can exchange help for free on our community forum at https://caddy.community. Remember that people give help out of their spare time and good will. The best way to get help is to give it first!

Please use our issue tracker only for bug reports and feature requests, i.e. actionable development items (support questions will usually be referred to the forums).

About

Matthew Holt began developing Caddy in 2014 while studying computer science at Brigham Young University. (The name "Caddy" was chosen because this software helps with the tedious, mundane tasks of serving the Web, and is also a single place for multiple things to be organized together.) It soon became the first web server to use HTTPS automatically and by default, and now has hundreds of contributors and has served trillions of HTTPS requests.

The name "Caddy" is trademarked. The name of the software is "Caddy", not "Caddy Server" or "CaddyServer". Please call it "Caddy" or, if you wish to clarify, "the Caddy web server". Caddy is a registered trademark of Stack Holdings GmbH.

- *Project on X: @caddyserver*
- *Author on X: @mholt6*

Caddy is a project of ZeroSSL, an HID Global company.

Debian package repository hosting is graciously provided by Cloudsmith. Cloudsmith is the only fully hosted