Caddy Web server, auto HTTPS

START HERE

`caddy` **`run`**
Run in the foreground

`caddy` **`validate`**
Check config without running

`caddy` **`start`**
Run in the background

`caddy` **`fmt --overwrite`**
Tidy the Caddyfile

`caddy` **`reload`**
Apply a changed config

`caddy` **`file-server --browse`**
Serve this folder

`caddy` **`stop`**
Stop gracefully

`caddy` **`reverse-proxy --from :80 --to :8080`**
Quick proxy

Caddyfile human-friendly config → `adapt` → **JSON config** native format → `load` / `reload` → **Admin API** localhost:2019 → `applies` → **Running server** auto HTTPS

CONCEPTS

With a Caddyfile in the current directory, caddy run loads it with no flags.

A site address with a real domain gets a public certificate automatically; localhost gets a locally trusted one.

The one-liner file-server and reverse-proxy commands disable the admin API, so caddy reload cannot touch them.

RUN AND CONTROL

`caddy` **`run`**
Run in the foreground
Loads ./Caddyfile if present; logs go to stderr.

`caddy` **`start`**
Run in the background

`caddy` **`stop`**
Stop the running instance gracefully

`caddy` **`reload`**
Apply the config to the running instance
Zero downtime; a failing config leaves the old one running.

`caddy` **`run --config`** *`<file>`*
Run with a specific config file

`caddy` **`run --watch`**
Reload automatically when the config changes
Meant for local development only.

`caddy` **`version`**
Print the version

CONFIG TOOLS

`caddy` **`validate --config`** *`<file>`*
Load and provision without running

CONFIG TOOLS (cont.)

`caddy` **`fmt --overwrite`** *`<file>`*
Format the file in place

`caddy` **`adapt --config`** *`<file>`*
Convert a Caddyfile to JSON

`caddy` **`run --config`** *`<file>`* **`--adapter caddyfile`**
Pick the config adapter by name
The caddyfile adapter is the default for a file named Caddyfile.

QUICK SERVERS

`caddy` **`file-server`**
Serve the current directory

`caddy` **`file-server --browse`**
Serve with a directory listing

`caddy` **`file-server --root`** *`<dir>`* **`--listen :`***`<port>`*
Serve a folder on a port

`caddy` **`file-server --access-log`**
Log every request

`caddy` **`reverse-proxy --from`** *`<addr>`* **`--to`** *`<backend>`*
Proxy requests to a backend

`caddy` **`reverse-proxy --change-host-header --to`** *`<url>`*
Send the backend's own Host header

`caddy` **`respond --status`** *`<code>`* **`"`***`<body>`***`"`**
Run a fixed-response test server

CADDYFILE BASICS

`example.com { ... }`
Site block with automatic HTTPS

`:8080 { ... }`
Site block on plain HTTP, port 8080

`http://example.com { ... }`
Explicit HTTP, no auto-HTTPS

CADDYFILE BASICS (cont.)

`{ email` *`<addr>`* **`}`**
Global options block, must come first

`reverse_proxy` *`<backend>`*
Send requests to a backend

`file_server`
Serve static files from the site root

`root *` *`<dir>`*
Set the document root for all paths

`encode zstd gzip`
Compress responses

`respond "`*`<text>`***`"`** *`<status>`*
Send a fixed response

`redir` *`<to>`* **`permanent`**
Redirect to another URL

`try_files {path} /index.html`
Single-page app fallback

`handle /api/* { ... }`
Exclusive block for a path prefix

`handle_path /api/* { ... }`
Same, but strips the prefix first

`header` *`<Name>`* *`<value>`*
Set a response header

`basic_auth {` *`<user>`* *`<hash>`* **`}`**
Require a username and password

`log`
Enable access logs for the site

`@`*`<name>`* **`{ ... }`**
Define a named request matcher

`(`*`<name>`***`) { ... }`** **`/ import`** *`<name>`*
Reusable snippet and its use

`{$VAR} / {env.VAR}`
Environment variable at load or run time
{$VAR} is substituted when the file is parsed; {env.VAR} per request.

CERTIFICATES AND PASSWORDS

`caddy` **`trust`**
Install the local CA in system trust stores

`caddy` **`hash-password`**
Prompt for a password and print its hash
Paste the result into basic_auth. Default algorithm is bcrypt.

RECIPES

Static site with SPA fallback

```
# Caddyfile
example.com {
  root * /var/www/site
  encode zstd gzip
  try_files {path} /index.html
  file_server
}
```

Proxy an API, serve the rest

```
# Caddyfile
example.com {
  handle /api/* {
    reverse_proxy
localhost:3000
  }
  handle {
    root * /var/www/site
    file_server
  }
}
```

Password-protect a site

```
caddy hash-password
# Caddyfile: paste the hash
example.com {
  basic_auth {
    alice <hash>
  }
  reverse_proxy localhost:8080
}
```

Edit, check, apply

```
caddy fmt --overwrite Caddyfile
caddy validate --config
Caddyfile
caddy reload --config Caddyfile
```