Caddy Web Server Cheatsheet

A quick reference guide to the Caddy web server, covering its configuration syntax, common directives, and usage examples.

Caddyfile Basics

Caddyfile Structure

A Caddyfile defines the configuration for one or more sites.

- **Site Addresses:** The starting point for each site definition.
- **Directives:** Instructions for how Caddy should handle requests to that site.
- **Blocks:** Group directives together to apply them conditionally or configure sub-features.

Caddyfile is case-insensitive.
Whitespace is generally ignored.

Site Addresses

`example.com`	Matches requests to `example.com`.
`example.com:8080`	Matches requests to `example.com` on port 8080.
`*.example.com`	Matches requests to any subdomain of `example.com`.
`http://example.com`	Matches HTTP requests to `example.com` (usually redirects to HTTPS).
`https://example.com`	Matches HTTPS requests to `example.com`.
`localhost`	Matches requests to `localhost`.

Basic Directives

`root * /var/www`	Sets the root directory for serving files.
`file_server`	Enables the file server to serve static files.
`encode gzip`	Enables Gzip compression for responses.
`log`	Configures logging.
`tls self_signed`	Uses a self-signed certificate for TLS (not recommended for production).
`reverse_proxy localhost:9000`	Proxies requests to a backend server.

Common Directives

Redirection

`redir /old /new`	Redirects `/old` to `/new`.
`redir /old /new 301`	Permanent redirect (301 status code).
`redir /old https://example.com/new 302`	Temporary redirect (302 status code) to a different domain.
`redir /.well-known/acme-challenge/* https://acme-staging.example.com{path}`	Redirects ACME challenge requests to a staging server.

Error Handling

`handle_errors { respond "An error occurred" 500 }`	Handles errors and returns a 500 error with a custom message.
`handle_errors { reverse_proxy localhost:9000 }`	Proxies error requests to another server.
`handle_errors { rewrite /error.html file_server }`	Serves a custom error page.

Header Manipulation

`header { Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" }`	Sets the Strict-Transport-Security header for HTTPS.
`header -Server`	Removes the Server header.
`header /api/* { Access-Control-Allow-Origin "*" }`	Sets CORS headers for the `/api/*` path.
`header +X-Custom-Header "Custom Value"`	Adds a custom header

Advanced Configuration

Reverse Proxy

`reverse_proxy /api/* localhost:8080` - Proxies requests to `/api/*` to a backend server on localhost:8080.

`reverse_proxy https://backend.example.com { header_up Host {host} header_up X-Real-IP {remote_host} }` - Proxies requests to another domain, preserving the original host and client IP.

`reverse_proxy /metrics 192.168.1.10:9100` - Proxies to an internal ip address

Load Balancing

`reverse_proxy localhost:8080 localhost:8081`	Basic load balancing between two backend servers.
`reverse_proxy { to localhost:8080 localhost:8081 policy round_robin }`	Load balancing with a specific policy (round_robin, first, random, least_conn, header).
`reverse_proxy { to 192.168.1.10:8080 192.168.1.11:8080 health_uri /healthz health_interval 10s }`	Configures health checks for backend servers.

TLS Configuration

`tls internal`	Uses Caddy's internal CA for issuing certificates (for development).
`tls { dns cloudflare <API_TOKEN> }`	Uses Cloudflare DNS for ACME DNS challenge.
`tls /path/to/cert.pem /path/to/key.pem`	Specifies the paths to a custom certificate and key.
`tls { protocols tls1.2 tls1.3 }`	Specifies the TLS protocol versions.

Caddy CLI

Basic Commands

`caddy version`	Displays the Caddy version.
`caddy start`	Starts Caddy in the background.
`caddy stop`	Stops Caddy.
`caddy reload`	Reloads the Caddy configuration without downtime.
`caddy adapt`	Converts a Caddyfile to JSON configuration.

Configuration Management

`caddy file-server`	Starts a simple file server.
`caddy run`	Starts Caddy in the foreground (useful for debugging).
`caddy environ`	Prints Caddy's environment variables.

Service Management (systemd)

Caddy can be managed as a systemd service. Ensure the Caddyfile is in the correct location (e.g., `/etc/caddy/Caddyfile`).

`sudo systemctl start caddy` - Starts the Caddy service.

`sudo systemctl stop caddy` - Stops the Caddy service.

`sudo systemctl reload caddy` - Reloads the Caddy configuration.

`sudo systemctl status caddy` - Checks the status of the Caddy service.