Reverse proxy quick-start - Caddy Documentation

caddyserver.com | Caddy Web Server | saved 21 Aug 2026
https://caddyserver.com/docs/quick-starts/reverse-proxy

This guide will show you how to get a production-ready reverse proxy with or without HTTPS up and running quickly.

Prerequisites:

- Basic terminal / command line skills
- caddy in your PATH
- A running backend process to proxy to

This tutorial assumes that you have a backend HTTP service running at 127.0.0.1:9000. These commands are for Linux, but the same principles apply to other operating systems.

You can get a simple reverse proxy running without a config file, or you can use a config file for more flexibility and control.

Command line

To start a plaintext HTTP proxy from port 2080 to port 9000 on your machine:

```
caddy reverse-proxy --from :2080 --to :9000
```

Then try it:

```
curl -v 127.0.0.1:2080
```

The reverse-proxy command is intended for quick and easy reverse proxies. (You can use it in production if your requirements are simple.)

Caddyfile

In the current working directory, create a file called Caddyfile with these contents:

```
:2080

reverse_proxy :9000
```

That config file is roughly equivalent to the caddy reverse-proxy command above.

Then, from the same directory, run:

```
caddy run
```

Then try your proxy:

```
curl -v 127.0.0.1:2080
```

Reverse proxy quick-start - Caddy Documentation

caddyserver.com | Caddy Web Server | saved 21 Aug 2026
https://caddyserver.com/docs/quick-starts/reverse-proxy

If you change the Caddyfile, make sure to reload Caddy.

This was a simple example. You can do a lot more with the reverse_proxy directive.

HTTPS from client to proxy

Caddy will serve your proxy over HTTPS automatically and by default if it knows the hostname (domain name). The caddy reverse-proxy command will default to localhost if you omit the --from flag, or you can replace the first line of your Caddyfile with the domain name of the proxy.

- If you use localhost or any domain ending in .localhost, Caddy will use an auto-renewing self-signed certificate. The first time you do this, you may need to enter a password as Caddy attempts to install its CA's root certificate into your trust store.
- If you use any other domain name, Caddy will attempt to get a publicly-trusted certificate; make sure your DNS records point to your machine and that ports 80 and 443 are open to the public and directed toward Caddy.

If you don't specify a port, Caddy defaults to 443 for HTTPS. In that case you will also need permission to bind to low ports. A couple ways to do this on Linux:

- Run as root (e.g. sudo -E).
- Or run sudo setcap cap_net_bind_service=+ep $(which caddy) to give Caddy this specific capability.

Here's the most basic caddy reverse-proxy command that gives you HTTPS:

```
caddy reverse-proxy --to :9000
```

Then try it:

```
curl -v https://localhost
```

You can customize the hostname using the --from flag:

```
caddy reverse-proxy --from example.com --to :9000
```

If you don't have permission to bind to low ports, you can proxy from a higher port:

```
caddy reverse-proxy --from example.com:8443 --to :9000
```

If you're using a Caddyfile, simply change the first line to your domain name, for example:

```
example.com

reverse_proxy :9000
```

Reverse proxy quick-start - Caddy Documentation

caddyserver.com | Caddy Web Server | saved 21 Aug 2026
https://caddyserver.com/docs/quick-starts/reverse-proxy

HTTPS from proxy to backend

Caddy can also proxy using HTTPS between itself and the backend if the backend supports TLS. Just use https:// in your backend address:

```
caddy reverse-proxy --from :2080 --to https://localhost:9000
```

This requires that the backend's certificate is trusted by the system Caddy is running on. (Caddy doesn't trust self-signed certificates unless explicitly configured to do so.)

Of course, you can do HTTPS on both ends too:

```
caddy reverse-proxy --from example.com --to https://example.com:9000
```

This serves HTTPS from client to proxy, and from proxy to backend.

If the hostname you're proxying to is different than the one you're proxying from, you will need to use the --change-host-header flag:

```
caddy reverse-proxy \
 --from example.com \
 --to https://localhost:9000 \
 --change-host-header
```

By default, Caddy passes all HTTP headers through unchanged, including Host, and Caddy derives the TLS ServerName from the Host header. The --change-host-header resets the Host header to that of the backend so that the TLS handshake can complete successfully. In the example above, it would be changed from example.com to localhost:9000 (and localhost would be used in the TLS handshake).