

Middlewares

- [Overview Middlewares](#)
- [HTTP Middlewares Overview](#)
- [TCP Middlewares Overview](#)

Overview Middlewares

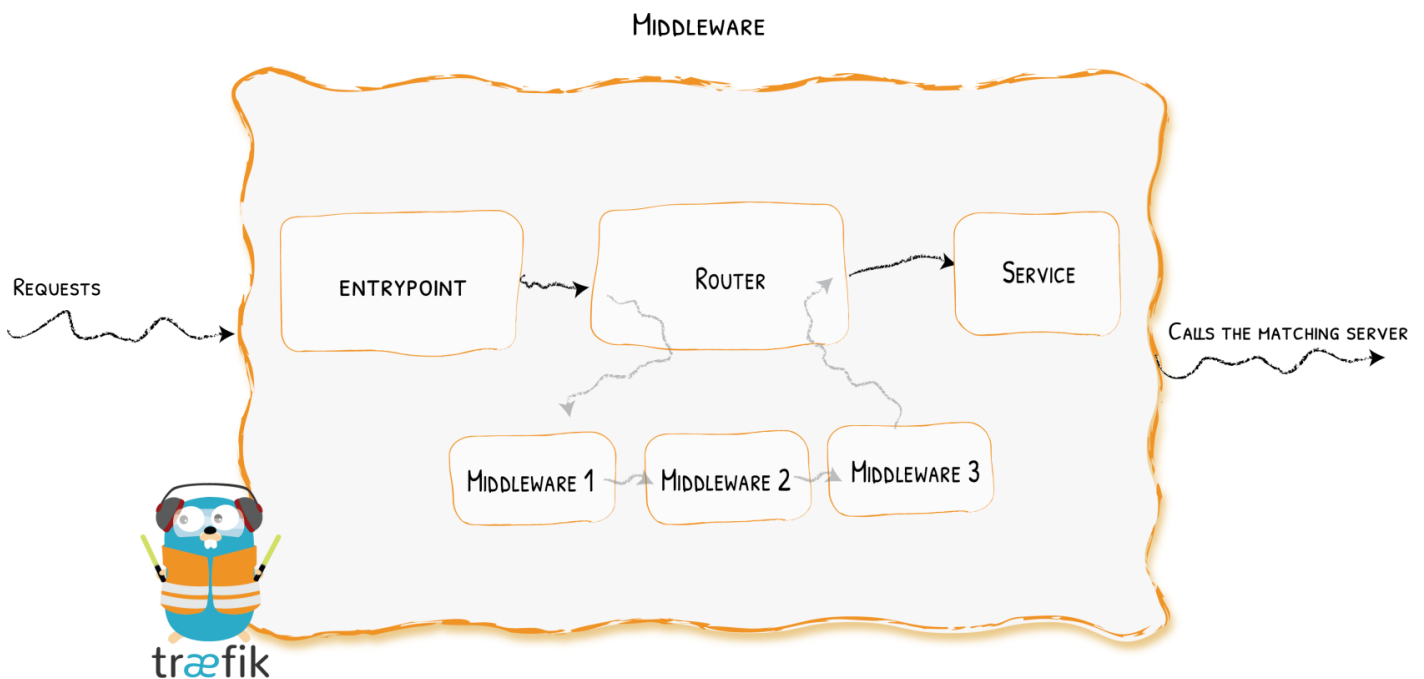


Attached to the routers, pieces of middleware are a means of tweaking the requests before they are sent to your service (or before the answer from the services are sent to the clients).

There are several available middleware in Traefik, some can modify the request, the headers, some are in charge of redirections, some add authentication, and so on.

Middlewares that use the same protocol can be combined into chains to fit every scenario.

Middlewares can be attached to Routers (specific Services) or also added to an Entrypoint. So every Service using this Entrypoint flows through that Middleware.



Available Middlewares

Traefik provides several official middlewares by default. Refer to the documentation for the full list:

- [HTTP Middlewares](#)
- [TCP Middlewares](#)

Additionally, you can explore the [Plugin Catalog](#), where community-driven middlewares are published.

Configuration Example directly in a `docker-compose.yml`

```
# As a Docker Label
whoami:
  # A container that exposes an API to show its IP address
  image: traefik/whoami
  labels:
    # Create a middleware named `foo-add-prefix`
    - "traefik.http.middlewares.foo-add-prefix.addprefix.prefix=/foo"
    # Apply the middleware named `foo-add-prefix` to the router named `router1`
    - "traefik.http.routers.router1.middlewares=foo-add-prefix@docker"
```

Configuration Example in an external `dynamic.yml`

```
http:
  routers:
    router1:
      service: myService
      middlewares:
        - "foo-add-prefix"
```

```
rule: "Host(`example.com`)"
```

```
middlewares:
```

```
foo-add-prefix:
```

```
addPrefix:
```

```
prefix: "/foo"
```

```
services:
```

```
service1:
```

```
loadBalancer:
```

```
servers:
```

```
- url: "http://127.0.0.1:80"
```

Configuration Example in a `dynamic.toml`

```
[http.routers]
```

```
[http.routers.router1]
```

```
service = "myService"
```

```
middlewares = ["foo-add-prefix"]
```

```
rule = "Host(`example.com`)"
```

```
[http.middlewares]
```

```
[http.middlewares.foo-add-prefix.addPrefix]
```

```
prefix = "/foo"
```

```
[http.services]
```

```
[http.services.service1]
```

```
[http.services.service1.loadBalancer]
```

```
[[http.services.service1.loadBalancer.servers]]
```

```
url = "http://127.0.0.1:80"
```

HTTP Middlewares Overview

HTTP Middlewares in Traefik let you modify requests and responses on the fly. You can configure them in multiple ways (Docker labels, TOML, YAML, etc.).

Configuration Example in a docker-compose.yaml

Configuration Example in a docker-compose.yaml

```
# As a Docker Label
whoami:
  # A container that exposes an API to show its IP address
  image: traefik/whoami
  labels:
    # Create a middleware named `foo-add-prefix`
    - "traefik.http.middlewares.foo-add-prefix.addprefix.prefix=/foo"
    # Apply the middleware named `foo-add-prefix` to the router named `router1`
    - "traefik.http.routers.router1.middlewares=foo-add-prefix@docker"
```

Configuration Example in a dynamic.toml

Configuration Example in a dynamic.toml

```
[http.routers]
[http.routers.router1]
  service = "service1"
  middlewares = ["foo-add-prefix"]
  rule = "Host(`example.com`)"
```

```
[http.middlewares]
[http.middlewares.foo-add-prefix.addPrefix]
  prefix = "/foo"

[http.services]
[http.services.service1]
[http.services.service1.loadBalancer]

[[http.services.service1.loadBalancer.servers]]
  url = "http://127.0.0.1:80"
```

Configuration Example in a dynamic.yml

Configuration Example in a dynamic.yml

```
http:
  routers:
    router1:
      service: service1
      middlewares:
        - "foo-add-prefix"
      rule: "Host(`example.com`)"

  middlewares:
    foo-add-prefix:
      addPrefix:
        prefix: "/foo"

  services:
    service1:
      loadBalancer:
        servers:
          - url: "http://127.0.0.1:80"
```

Available HTTP Middlewares

Middleware	Purpose	Area
<u>AddPrefix</u>	Adds a Path Prefix	Path Modifier
<u>BasicAuth</u>	Adds Basic Authentication	Security, Authentication
<u>Buffering</u>	Buffers the request/response	Request Lifecycle
<u>Chain</u>	Combines multiple pieces of middleware	Misc
<u>CircuitBreaker</u>	Prevents calling unhealthy services	Request Lifecycle
<u>Compress</u>	Compresses the response	Content Modifier
<u>ContentType</u>	Handles Content-Type auto-detection	Misc
<u>DigestAuth</u>	Adds Digest Authentication	Security, Authentication
<u>Errors</u>	Defines custom error pages	Request Lifecycle
<u>ForwardAuth</u>	Delegates Authentication	Security, Authentication
<u>Headers</u>	Adds / Updates headers	Security
<u>IPAllowList</u>	Limits the allowed client IPs	Security, Request lifecycle
<u>InFlightReq</u>	Limits the number of simultaneous connections	Security, Request lifecycle
<u>PassTLSClientCert</u>	Adds Client Certificates in a Header	Security
<u>RateLimit</u>	Limits the call frequency	Security, Request lifecycle
<u>RedirectScheme</u>	Redirects based on scheme	Request lifecycle
<u>RedirectRegex</u>	Redirects based on regex	Request lifecycle
<u>ReplacePath</u>	Changes the path of the request	Path Modifier
<u>ReplacePathRegex</u>	Changes the path of the request	Path Modifier
<u>Retry</u>	Automatically retries in case of error	Request lifecycle

Middleware	Purpose	Area
StripPrefix	Changes the path of the request	Path Modifier
StripPrefixRegex	Changes the path of the request	Path Modifier

For even more options, check out the [community-contributed plugins](#) in the plugin catalog.

TCP Middlewares Overview

TCP Middlewares in Traefik let you manage connections on the fly. You can configure them in multiple ways (Docker labels, TOML, YAML, etc.).

Configuration Example in a `docker-compose.yaml`

Configuration Example in a `docker-compose.yaml`

```
# As a Docker Label
whoami:
  # A container that exposes an API to show its IP address
  image: traefik/whoami
  labels:
    # Create a middleware named `foo-ip-allowlist`
    - "traefik.tcp.middlewares.foo-ip-allowlist.ipallowlist.sourcerange=127.0.0.1/32, 192.168.1.7"
    # Apply the middleware named `foo-ip-allowlist` to the router named `router1`
    - "traefik.tcp.routers.router1.middlewares=foo-ip-allowlist@docker"
```

Configuration Example in a `dynamic.yml`

Configuration Example in a `dynamic.yml`

```
tcp:
  routers:
    router1:
      service: myService
      middlewares:
        - "foo-ip-allowlist"
```

```
rule: "Host(`example.com`)"
```

```
middlewares:
```

```
foo-ip-allowlist:
```

```
ipAllowList:
```

```
sourceRange:
```

```
- "127.0.0.1/32"
```

```
- "192.168.1.7"
```

```
services:
```

```
service1:
```

```
loadBalancer:
```

```
servers:
```

```
- address: "10.0.0.10:4000"
```

```
- address: "10.0.0.11:4000"
```

Available TCP Middlewares

Middleware	Purpose	Area
<u>InFlightConn</u>	Limits the number of simultaneous connections.	Security, Request lifecycle
<u>IPAllowList</u>	Limit the allowed client IPs.	Security, Request lifecycle