Traefik Edge router and proxy

START HERE

`traefik --configFile=<file>`
Start with a static config

`traefik.http.routers.<r>.rule`
Label: match requests

`--providers.docker=true`
Discover Docker containers

`traefik.http.services.<s>.loadbalancer.server.port`
Label: backend port

`--entryPoints.web.address=:80`
Open a listening port

`Host(`a.example.com`)`
Route by hostname

`--api.dashboard=true`
Enable the dashboard

`traefik healthcheck`
Probe a running instance

EntryPoint port :80 / :443 → `rule matches` → **Router** rule + priority → `runs` → **Middlewares** auth, redirect, headers → `forwards` → **Service** load balancer → `to` → **Backend** container / server

CONCEPTS

Static config (entrypoints, providers, resolvers) is read at start; dynamic config (routers, services, middlewares) reloads live.

Use one static method only: file, CLI flags or environment variables. Mixing them gives unexpected results.

Rules use backticks; single quotes are rejected. In YAML or labels, quote the whole value.

RUN AND STATIC CONFIG

`traefik --configFile=<file>`
Start with a config file
Without the flag it looks for traefik.yml, .yaml or .toml in default folders.

`traefik --entryPoints.<name>.address=<addr>`
Define an entry point, e.g. :80

`traefik --providers.docker=true`
Enable the Docker provider

`traefik --providers.file.directory=<dir>`
Load dynamic config from a folder

`traefik --api.dashboard=true`
Serve the dashboard

`traefik --log.level=DEBUG`
Set the log level

`traefik --providers.docker.exposedByDefault=false`
Expose only labelled containers

`traefik --providers.file.watch=true`
Reload file config on change

DOCKER LABELS

`traefik.enable=true`
Include this container

`traefik.http.routers.<r>.rule=Host(`<host>`)`
Route a hostname to it

`traefik.http.routers.<r>.entrypoints=<ep>`
Listen on this entry point

DOCKER LABELS (cont.)

`traefik.http.services.<s>.loadbalancer.server.port=<port>`
Backend port inside the container

`traefik.http.routers.<r>.tls=true`
Serve this router over TLS

`traefik.http.routers.<r>.tls.certresolver=<res>`
Get a certificate from a resolver

`traefik.http.routers.<r>.middlewares=<m>`
Attach middlewares, comma separated

`traefik.docker.network=<net>`
Network Traefik uses to reach it

`traefik.http.services.<s>.loadbalancer.healthcheck.path=<path>`
Probe the backend on a path

`traefik.http.middlewares.<m>.<type>.<opt>=<val>`
Define a middleware inline

ROUTER RULES

`Host(`<domain>`)`
Match a hostname

`PathPrefix(`/<prefix>`)`
Match a path prefix

`Path(`/<path>`)`
Match an exact path

`Host(`a`) && PathPrefix(`/api`)`
Combine with AND

`Host(`a`) || Host(`b`)`
Combine with OR

`HostRegexp(`^[a-z]+\.example\.com$`)`
Match a hostname by regex
HostRegexp cannot carry a TLS option; prefer Host with a wildcard.

`ClientIP(`10.0.0.0/8`)`
Match the client address or CIDR
Uses the socket peer, not X-Forwarded-For.

Router priority

Case	Result
Default	Longer rule string wins
priority: <n>	Higher number wins
Same priority	Provider order decides
Wildcard Host	Matches one label only

MIDDLEWARES

`redirectscheme.scheme=https`
Redirect HTTP to HTTPS

`stripprefix.prefixes=/<prefix>`
Remove a path prefix

`basicauth.users=<user>:<hash>`
Password-protect a route
Generate the hash with htpasswd; double each $ in compose files.

`ratelimit.average=<n>`
Limit requests per second

`ipallowlist.sourcerange=<cidr>`
Allow only listed clients

`forwardauth.address=<url>`
Delegate auth to a service

TLS AND ACME

`--certificatesresolvers.<res>.acme.email=<addr>`
Account email

`--certificatesresolvers.<res>.acme.storage=<file>`
Where certs are stored
Make it a persistent volume; the file must be mode 600.

`--certificatesresolvers.<res>.acme.tlschallenge=true`
Use the TLS-ALPN challenge

`--certificatesresolvers.<res>.acme.httpchallenge.entrypoint=web`
Use the HTTP challenge

`--certificatesresolvers.<res>.acme.dnschallenge.provider=<p>`
Use a DNS challenge (wildcards)

`--certificatesresolvers.<res>.acme.caserver=<url>`
Use another ACME server
Point at the Let's Encrypt staging directory while testing to avoid rate limits.

`--entryPoints.web.http.redirections.entryPoint.to=websecure`
Redirect a whole entry point

FILE PROVIDER

`http.routers.<r>.rule`
Router in dynamic YAML

`http.routers.<r>.service`
Which service it forwards to

FILE PROVIDER (cont.)

`http.services.<s>.loadBalancer.servers[].url`
Backend address in a service

`api@internal`
Service name of the dashboard

HEALTH AND DEBUGGING

`traefik healthcheck`
Check /ping, exit 0 if healthy
Needs ping enabled; use it as the container HEALTHCHECK.

`--ping=true`
Enable the /ping endpoint

`curl http://localhost:8080/api/rawdata`
Dump the live routers and services

RECIPES

Docker Compose: HTTPS service

```
services:
  traefik:
    image: traefik:v3.7
    command:
      - --providers.docker.
exposedByDefault=false
      - --
entryPoints.web.address=:80
      - --entryPoints.
websecure.address=:443
      - --
certificatesresolvers.le.acme.
tlschallenge=true
      - --
certificatesresolvers.le.acme.
email=me@example.com
      - --
certificatesresolvers.le.acme.
storage=/le/acme.json
    ports: ["80:80", "443:443"]
    volumes:
      - /var/run/docker.sock:/
var/run/docker.sock:ro
```

Labels on the app container

```
labels:
  - traefik.enable=true
  - traefik.http.routers.app.
rule=Host(`app.example.com`)
  - traefik.http.routers.app.
entrypoints=websecure
  - traefik.http.routers.app.
tls.certresolver=le
  - traefik.http.services.app.
loadbalancer.server.port=8080
```

traefik.yml: static config

```
entryPoints:
  web:
    address: :80
    http:
      redirections:
        entryPoint: {to:
websecure}
  websecure:
    address: :443
providers:
  file: {directory: /etc/
traefik/dynamic, watch: true}
api: {dashboard: true}
```