# Traefik Edge router and proxy

## CONCEPTS

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

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

**`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

**`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

## MIDDLEWARES

**`redirectscheme.scheme=https`**
Redirect HTTP to HTTPS

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

**`basicauth.users=`***`<user>`***`:`***`<hash>`*
Password-protect a route

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

## TLS AND ACME

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

**`--certificatesresolvers.`***`<res>`***`.acme.storage=`***`<file>`*
Where certs are stored

**`--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

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

## FILE PROVIDER

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

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

## HEALTH AND DEBUGGING

**`traefik healthcheck`**
Check /ping, exit 0 if healthy