HAProxy Load balancer and proxy

START HERE

`haproxy -c -f <file>`
Check the config

`kill -USR2 <master-pid>`
Reload the config

`haproxy -f <file>`
Run in foreground

`echo "show info" | socat /run/haproxy.sock stdio`
Process status

`haproxy -D -f <file>`
Run as daemon

`echo "show stat" | socat /run/haproxy.sock stdio`
Per-server stats

`haproxy -W -f <file>`
Master-worker mode

`echo "disable server <be>/<srv>" | socat /run/haproxy.sock stdio`
Drain for maintenance

frontend bind, ACLs → `use_backend` → **backend** balance, checks → `balance` → **server** app instance

A request flows through frontend rules to a backend

CONCEPTS

Always run haproxy -c before a reload: a bad config keeps the old process serving.

global process settings
- **defaults** shared timeouts, mode
  - **frontend / backend** listen = both in one

Sections of haproxy.cfg

A listen section is a frontend and a backend in one; handy for TCP or the stats page.

Runtime changes via the socket are lost on reload; persist them in the config.

PROCESS CONTROL

`haproxy -c -f <file>`
Validate config and exit

`haproxy -f <file>`
Start with a config file

`haproxy -f <dir>`
Load all .cfg files in a directory

`haproxy -D -f <file>`
Fork into the background

`haproxy -W -f <file>`
Master-worker mode

`haproxy -db -f <file>`
Foreground, no master-worker

`haproxy -d -f <file>`
Debug mode, show events

`haproxy -p <pidfile> -f <file>`
Write PIDs to a file

`haproxy -sf <pid>`
Reload; old process stops gracefully

`haproxy -st <pid>`
Reload; old process killed at once

`haproxy -v`
Print version and build date

`haproxy -vv`
Build options and libraries

`haproxy -S <bind> -W -f <file>`
Open a master CLI socket

`kill -USR2 <master-pid>`
Reload in master-worker mode

`kill -USR1 <pid>`
Graceful stop

`kill -TERM <pid>`
Hard stop

RUNTIME API

`stats socket /run/haproxy.sock mode 600 level admin`
Enable the socket (global section)

`echo "<command>" | socat /run/haproxy.sock stdio`
Run one command via socat

`echo "show info;show stat" | socat /run/haproxy.sock stdio`
Several commands, semicolon-separated

`socat /run/haproxy.sock readline`
Interactive session

`prompt`
Toggle interactive mode

`help`
List the available commands

`show info`
Version, uptime, load, limits

`show stat`
CSV stats for all proxies

`show stat json`
Stats as JSON

`show servers state <backend>`
Server states and weights

`show backend`
List backends

`show errors`
Last request/response errors

SERVERS AND LIMITS

`disable server <be>/<srv>`
Put server in maintenance

`enable server <be>/<srv>`
Bring server back

`set server <be>/<srv> state drain`
Stop new sessions, keep persisted

`set server <be>/<srv> state ready`
Normal mode

`set server <be>/<srv> weight <n>[%]`
Change load share

`get weight <be>/<srv>`
Show current and initial weight

`set server <be>/<srv> addr <ip> [port <p>]`
Change server address

SERVERS AND LIMITS (cont.)

`set maxconn global <n>`
Change global maxconn

TABLES, ACLS AND MAPS

`show table`
List stick-tables

`show table <name>`
Dump entries of one table

`show acl`
List ACL files and IDs

`add acl <acl> <pattern>`
Add an ACL entry

`del acl <acl> <key>`
Remove an ACL entry

`show map`
List maps

MASTER CLI AND TLS

`reload`
Synchronous reload via master CLI

`show proc`
List master and worker processes

CONFIG BASICS

`global`
Process-wide settings section

`defaults`
Shared mode and timeouts

`frontend <name>`
Where clients connect

`backend <name>`
Pool of servers

`listen <name>`
Frontend and backend in one

`bind :<port>`
Accept connections on a port

`mode http`
Layer 7 proxying

`mode tcp`
Layer 4 proxying

`server <name> <ip>:<port> check`
Add a checked server

`balance roundrobin`
Rotate across servers

`balance leastconn`
Fewest connections wins

`default_backend <name>`
Fallback backend

CONFIG BASICS (cont.)

`use_backend <name> if <acl>`
Route by condition

`acl <name> <criterion> <value>`
Define a named condition

`http-request deny if <acl>`
Reject matching requests

`http-request redirect scheme https unless { ssl_fc }`
Force HTTPS

`http-request set-header <name> <value>`
Set a request header

`option forwardfor`
Add X-Forwarded-For

`option httpchk`
HTTP health checks

`timeout client|server|connect <t>`
Set timeouts

`maxconn <n>`
Connection limit

`stats uri <path>`
Serve the stats page

`stats auth <user>:<pass>`
Protect the stats page

`option httplog`
Detailed HTTP logs

RECIPES

Minimal HTTP load balancer

```
frontend web
  bind :80
  default_backend app

backend app
  balance roundrobin
  option httpchk GET /health
  server a1 10.0.0.11:8080 check
  server a2 10.0.0.12:8080 check
```

Route by host and redirect to HTTPS

```
frontend web
  bind :80
  bind :443 ssl crt /etc/haproxy/site.pem
  http-request redirect scheme https unless { ssl_fc }
  acl is_api hdr(host) -i api.example.com
  use_backend api if is_api
  default_backend app
```

Stats page

```
listen stats
  bind 127.0.0.1:8404
  mode http
  stats enable
  stats uri /stats
  stats refresh 10s
```