Varnish HTTP cache

START HERE

`varnishd -a :80 -f <file.vcl>`
Start the cache

`varnishlog -g request`
Grouped log

`varnishadm vcl.load <name> <file>`
Compile a VCL

`varnishtop -i ReqURL`
Top URLs

`varnishadm vcl.use <name>`
Switch to it

`varnishadm ban "<expr>"`
Invalidate objects

`varnishadm backend.list`
Backend health

`varnishstat -1`
Counters once

CONCEPTS

Varnish is a caching reverse proxy: clients hit varnishd, which answers from cache or fetches from the backend. Behaviour is set in VCL, changed live with varnishadm.

Client HTTP request
↓ `vcl_recv`
Cache hit: serve object
↓ `miss`
Backend fetch, vcl_backend_response
↓ `store`
Client vcl_deliver

Only the active VCL handles new requests; older loaded VCLs stay for instant rollback until discarded.

RUN VARNISHD

`varnishd -a :<port> -f <file.vcl>`
Listen on a port with a VCL file

`varnishd -a :<port> -b <host>:<port>`
Quick start with a single backend

`varnishd -T localhost:6082 -S <secretfile>`
Open the management CLI

`varnishd -s malloc,<size>`
Cache in memory, e.g. malloc,1G

`varnishd -V`
Print the version

VARNISHADM

`varnishadm`
Interactive CLI to varnishd

`varnishadm <command>`
Run one CLI command

`varnishadm -T <host>:<port> -S <secretfile>`
Connect over the management port

`varnishadm help`
List all CLI commands

`varnishadm status`
Is the cache process running?

VCL MANAGEMENT

`varnishadm vcl.load <name> <file>`
Compile and load a VCL
Loading does not activate it. A compile error leaves the running VCL untouched.

`varnishadm vcl.use <name>`
Make a loaded VCL active

`varnishadm vcl.list`
List VCLs, state and temperature

`varnishadm vcl.show <name>`
Print the VCL source

`varnishadm vcl.discard <name>`
Unload a VCL

BANS AND PURGING

`varnishadm ban req.url ~ "<regex>"`
Invalidate by URL pattern

`varnishadm ban req.http.host == "<host>" && req.url ~ "<regex>"`
Combine conditions with &&

`varnishadm ban.list`
Show active bans

`curl -X PURGE <url>`
Purge one object via VCL
Needs a PURGE handler with return(purge) in vcl_recv, ideally ACL-protected.

BACKENDS AND HEALTH

`varnishadm backend.list`
List backends and health

`varnishadm backend.list -p`
Include probe details

`varnishadm backend.set_health <backend> sick`
Take a backend out of rotation

`varnishadm backend.set_health <backend> auto`
Return control to the probe

PARAMETERS

`varnishadm param.show`
List parameters and values

PARAMETERS (cont.)

`varnishadm param.show changed`
Only non-default parameters

`varnishadm param.set <param> <value>`
Change a parameter live

VARNISHSTAT

`varnishstat`
Live counters (curses)

`varnishstat -1`
Print all counters once

`varnishstat -1 -f MAIN.cache_hit -f MAIN.cache_miss`
Select counters (globs)

`MAIN.cache_hit, MAIN.cache_miss`
Compute the hit ratio

VARNISHLOG

`varnishlog`
Log every transaction

`varnishlog -g request`
Group by client request tree

`varnishlog -c`
Client side only

`varnishlog -b`
Backend side only

`varnishlog -q '<VSL query>'`
Filter transactions

`varnishlog -q 'ReqURL ~ "^/api"'`
Match by URL

VARNISHLOG (cont.)

`varnishlog -q 'RespStatus >= 500'`
Only server errors

VARNISHTOP AND FRIENDS

`varnishtop -i ReqURL`
Most requested URLs

`varnishncsa`
Apache-style access log

VCL ESSENTIALS

`vcl 4.1;`
Required first line

`backend default { .host = "<host>"; .port = "<port>"; }`
Declare a backend

`sub vcl_recv { ... }`
Request arrives; choose hash, pass, purge

`return (pass);`
Skip the cache for this request

`sub vcl_backend_response { set beresp.ttl = 5m; }`
Set cache lifetime

`sub vcl_deliver { ... }`
Edit the response to the client

`set beresp.grace = 1h;`
Serve stale while refetching

`unset req.http.Cookie;`
Drop cookies so pages cache

VCL ESSENTIALS (cont.)

`set resp.http.X-Cache = obj.hits > 0 ? "HIT" : "MISS";`
Expose hit status

Common VCL variables

Variable	Meaning
`req.url`	request path and query
`req.http.<h>`	request header
`bereq.*`	request sent to backend
`beresp.ttl`	object lifetime
`obj.hits`	times served from cache
`resp.http.<h>`	response header to client

RECIPES

Deploy a new VCL with rollback

```
varnishadm vcl.load v2 /etc/
varnish/v2.vcl
varnishadm vcl.use v2
# something wrong? roll back:
varnishadm vcl.use v1
varnishadm vcl.discard v2
```

Cache ratio at a glance

```
varnishstat -1 -f
MAIN.cache_hit -f
MAIN.cache_miss
# live and updating:
watch -n2 'varnishstat -1 -f
MAIN.cache_hit -f
MAIN.cache_miss'
```

Debug one URL end to end

```
varnishlog -g request -q
'ReqURL ~ "^/checkout"'
# in another terminal
curl -sI http://localhost/
checkout
```