# Varnish HTTP cache

## 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

## 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

## 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

**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.list**
Show active bans

## 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

**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 -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 '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

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

## 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
```