Varnish

Varnish Cache is a web application accelerator also known as a caching HTTP reverse proxy. You install it in front of any server that speaks HTTP and configure it to cache the contents. Varnish Cache is really, really fast. It typically speeds up delivery with a factor of 300 - 1000x, depending on your architecture.

Varnish for Wordpress

Install dulu varnish-nya:

```
sudo apt update && sudo apt install varnish
```

ada 2 berkas yang perlu diedit untuk mengubah port varnish

- /etc/default/varnish
- /lib/systemd/system/varnish.service

kita edit yang pertama dulu:

```
vim /etc/default/varnish
```

Ganti port 6081 menjadi 80 pada `DAEMON_OPTS` setelah opsi `-a` sehingga menjadi seperti berikut:

```
DAEMON_OPTS="-a :80 \
           -T localhost:6082 \
           -f /etc/varnish/default.vcl \
           -S /etc/varnish/secret \
           -s malloc,1024m"
```

Oiya, sesuaikan juga nilai untuk `malloc` dengan memori yang tersedia di server.

Selanjutnya sunting berkas systemd service berikut:

```
vim /lib/systemd/system/varnish.service
```

Mirip sih, ubah port 6081 menjadi port 80, pada bagian `ExecStart` sebagai berikut:

```
ExecStart=/usr/sbin/varnishd \
        -j unix,user=vcache \
        -F \
        -a :80 \
        -T localhost:6082 \
        -f /etc/varnish/default.vcl \
        -S /etc/varnish/secret \
        -s malloc,1024m
```

Berikutnya, backup berkas konfigurasi default

```
cd /etc/varnish/
mv default.vcl{,.orig}
vim default.vcl
```

Isikan berkas konfigurasi tersebut dengan teks berikut:

```
#
# This is an example VCL file for Varnish.
#
# It does not do anything by default, delegating control to the
# builtin VCL. The builtin VCL is called when there is no explicit
# return statement.
#
# See the VCL chapters in the Users Guide at
https://www.varnish-cache.org/docs/
# and https://www.varnish-cache.org/trac/wiki/VCLExamples for more examples.

# Marker to tell the VCL compiler that this VCL has been adapted to the
# new 4.0 format.
vcl 4.0;

import std;
import directors;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8080";
    #.max_connections = 300; # That's it

  #.probe = {
    #.url = "/"; # short easy way (GET /)
    # We prefer to only do a HEAD /
  #  .request =
  #    "HEAD / HTTP/1.1"
  #    "Host: localhost"
  #    "Connection: close"
  #    "User-Agent: Varnish Health Probe";

  #  .interval  = 5s; # check the health of each backend every 5 seconds
  #  .timeout   = 1s; # timing out after 1 second.
  #  .window    = 5;  # If 3 out of the last 5 polls succeeded the backend
is considered healthy, otherwise it will be marked as sick
  #  .threshold = 3;
  #}

  #.first_byte_timeout     = 300s;   # How long to wait before we receive a
first byte from our backend?
  #.connect_timeout        = 5s;     # How long to wait for a backend
```

```
connection?
  #.between_bytes_timeout  = 2s;     # How long to wait between bytes
received from our backend?

}

#acl purge {
  # ACL we'll use later to allow purges
#  "localhost";
#  "127.0.0.1";
#  "::1";
#}

sub vcl_recv {
    # Happens before we check if we have this in cache already.
    #
    # Typically you clean up the request here, removing cookies you don't
need,
    # rewriting the request, etc.

  #set req.backend_hint = vdir.backend(); # send all traffic to the vdir
director

  # Normalize the header, remove the port (in case you're testing this on
various TCP ports)
  #set req.http.Host = regsub(req.http.Host, ":[0-9]+", "");

  # Remove the proxy header (see https://httpoxy.org/#mitigate-varnish)
  unset req.http.proxy;

  # Normalize the query arguments
  set req.url = std.querysort(req.url);

  # Allow purging
  #if (req.method == "PURGE") {
  #  if (!client.ip ~ purge) { # purge is the ACL defined at the begining
      # Not from an allowed IP? Then die with an error.
  #    return (synth(405, "This IP is not allowed to send PURGE
requests."));
  #  }
    # If you got this stage (and didn't error out above), purge the cached
result
  #  return (purge);
  #}

  # Only deal with "normal" types
  if (req.method != "GET" &&
      req.method != "HEAD" &&
      req.method != "PUT" &&
      req.method != "POST" &&
      req.method != "TRACE" &&
```

```
      req.method != "OPTIONS" &&
      req.method != "PATCH" &&
      req.method != "DELETE") {
  #  /* Non-RFC2616 or CONNECT which is weird. */
  #  /*Why send the packet upstream, while the visitor is using a non-valid
HTTP method? */
    return (synth(404, "Non-valid HTTP method!"));
  }

  # Implementing websocket support
(https://www.varnish-cache.org/docs/4.0/users-guide/vcl-example-websockets.h
tml)
  if (req.http.Upgrade ~ "(?i)websocket") {
    return (pipe);
  }

  # Only cache GET or HEAD requests. This makes sure the POST requests are
always passed.
  if (req.method != "GET" && req.method != "HEAD") {
    return (pass);
  }

  # Some generic URL manipulation, useful for all templates that follow
  # First remove URL parameters used to track effectiveness of online
marketing campaigns
  if (req.url ~ "(\?|&)(utm_[a-z]+|gclid|cx|ie|cof|siteurl|fbclid)=") {
      set req.url = regsuball(req.url, "(utm_[a-
z]+|gclid|cx|ie|cof|siteurl|fbclid)=[-_A-z0-9+()%.]+&?", "");
      set req.url = regsub(req.url, "[?|&]+$", "");
  }

  # Strip hash, server doesn't need it.
  if (req.url ~ "\#") {
    set req.url = regsub(req.url, "\#.*$", "");
  }

  # Strip a trailing ? if it exists
  if (req.url ~ "\?$") {
    set req.url = regsub(req.url, "\?$", "");
  }

  # Some generic cookie manipulation, useful for all templates that follow
  # Remove the "has_js" cookie
  set req.http.Cookie = regsuball(req.http.Cookie, "has_js=[^;]+(; )?", "");

  # Remove any Google Analytics based cookies
  set req.http.Cookie = regsuball(req.http.Cookie, "__utm.=[^;]+(; )?", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "_ga=[^;]+(; )?", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "_gat=[^;]+(; )?", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "utmctr=[^;]+(; )?", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "utmcmd.=[^;]+(; )?",
```

```
"");
  set req.http.Cookie = regsuball(req.http.Cookie, "utmccn.=[^;]+(; )?",
"");

  # Remove DoubleClick offensive cookies
  set req.http.Cookie = regsuball(req.http.Cookie, "__gads=[^;]+(; )?", "");

  # Remove the Quant Capital cookies (added by some plugin, all __qca)
  set req.http.Cookie = regsuball(req.http.Cookie, "__qc.=[^;]+(; )?", "");

  # Remove the AddThis cookies
  set req.http.Cookie = regsuball(req.http.Cookie, "__atuv.=[^;]+(; )?",
"");

  # Remove a ";" prefix in the cookie if present
  set req.http.Cookie = regsuball(req.http.Cookie, "^;\s*", "");

  # Are there cookies left with only spaces or that are empty?
  if (req.http.cookie ~ "^\s*$") {
    unset req.http.cookie;
  }
  # Large static files are delivered directly to the end-user without
  # waiting for Varnish to fully read the file first.
  # Varnish 4 fully supports Streaming, so set do_stream in
vcl_backend_response()
  if (req.url ~
"^[^?]*\.(7z|avi|bz2|flac|flv|gz|mka|mkv|mov|mp3|mp4|mpeg|mpg|ogg|ogm|opus|r
ar|tar|tgz|tbz|txz|wav|webm|xz|zip)(\?.*)?$") {
    unset req.http.Cookie;
    return (hash);
  }

  # Send Surrogate-Capability headers to announce ESI support to backend
  set req.http.Surrogate-Capability = "key=ESI/1.0";

  if (req.http.Authorization) {
    # Not cacheable by default
    return (pass);
  }

  return (hash);

}

#sub vcl_pipe {
  # Called upon entering pipe mode.
  # In this mode, the request is passed on to the backend, and any further
data from both the client
  # and backend is passed on unaltered until either end closes the
connection. Basically, Varnish will
  # degrade into a simple TCP proxy, shuffling bytes back and forth. For a
```

```
connection in pipe mode,
  # no other VCL subroutine will ever get called after vcl_pipe.

  # Note that only the first request to the backend will have
  # X-Forwarded-For set.  If you use X-Forwarded-For and want to
  # have it set for all requests, make sure to have:
  # set bereq.http.connection = "close";
  # here.  It is not set by default as it might break some broken web
  # applications, like IIS with NTLM authentication.

  #set bereq.http.Connection = "Close";

  # Implementing websocket support
(https://www.varnish-cache.org/docs/4.0/users-guide/vcl-example-websockets.h
tml)
  #if (req.http.upgrade) {
  #  set bereq.http.upgrade = req.http.upgrade;
  #}

  #return (pipe);
#}

sub vcl_backend_response {
    # Happens after we have read the response headers from the backend.
    #
    # Here you clean the response headers, removing silly Set-Cookie headers
    # and other mistakes your backend does.
  # Pause ESI request and remove Surrogate-Control header
  #if (beresp.http.Surrogate-Control ~ "ESI/1.0") {
  #  unset beresp.http.Surrogate-Control;
  #  set beresp.do_esi = true;
  #}

  # Sometimes, a 301 or 302 redirect formed via Apache's mod_rewrite can
mess with the HTTP port that is being passed along.
  # This often happens with simple rewrite rules in a scenario where Varnish
runs on :80 and Apache on :8080 on the same box.
  # A redirect can then often redirect the end-user to a URL on :8080, where
it should be :80.
  # This may need finetuning on your setup.
  #
  # To prevent accidental replace, we only filter the 301/302 redirects for
now.
  if (beresp.status == 301 || beresp.status == 302) {
    set beresp.http.Location = regsub(beresp.http.Location, ":[0-9]+", "");
  }

  # Don't cache 50x responses
  if (beresp.status == 500 || beresp.status == 502 || beresp.status == 503
|| beresp.status == 504) {
    return (abandon);
```

```
  }

  # Set 2min cache if unset for static files
  if (beresp.ttl <= 0s || beresp.http.Set-Cookie || beresp.http.Vary == "*")
{
    set beresp.ttl = 120s; # Important, you shouldn't rely on this, SET YOUR
HEADERS in the backend
    set beresp.uncacheable = true;
    return (deliver);
  }

  # Allow stale content, in case the backend goes down.
  # make Varnish keep all objects for 6 hours beyond their TTL
  set beresp.grace = 6h;

  return (deliver);

}

sub vcl_deliver {
    # Happens when we have all the pieces we need, and are about to send the
    # response to the client.
    #
    # You can do accounting or modifying the final object here.
    #

  if (obj.hits > 0) { # Add debug header to see if it's a HIT/MISS and the
number of hits, disable when not needed
    set resp.http.X-Cache = "HIT";
  } else {
    set resp.http.X-Cache = "MISS";
  }

  # Please note that obj.hits behaviour changed in 4.0, now it counts per
objecthead, not per object
  # and obj.hits may not be reset in some cases where bans are in use. See
bug 1492 for details.
  # So take hits with a grain of salt
  set resp.http.X-Cache-Hits = obj.hits;

  # Remove some headers: PHP version
  unset resp.http.X-Powered-By;

  # Remove some headers: Apache version & OS
  unset resp.http.Server;
  unset resp.http.X-Drupal-Cache;
  unset resp.http.X-Varnish;
  unset resp.http.Via;
  unset resp.http.Link;
  unset resp.http.X-Generator;
```

```
  return (deliver);
}

sub vcl_fini {
  # Called when VCL is discarded only after all requests have exited the
VCL.
  # Typically used to clean up VMODs.

  return (ok);
}
```

jangan lupa ubah port web server dengan port 8080, misalnya di nginx seperti berikut:

```
server {
    listen 8080;
    server_name www.blog;

# other config goes here

}
```

restart kedua service:

```
nginx -t
systemctl daemon-reload
systemctl restart nginx
systemctl restart varnish
```

coba test dengan curl:

```
root@wp-dev-blog2:/etc/varnish# curl -I localhost
HTTP/1.1 200 OK
Date: Thu, 07 Jul 2022 18:20:21 GMT
Content-Type: text/html
last-modified: Thu, 07 Jul 2022 04:30:44 GMT
vary: Accept-Encoding
etag: W/"62c66174-17c8"
Cache-Control: private
Age: 0
X-Cache: MISS
X-Cache-Hits: 0
Accept-Ranges: bytes
Connection: keep-alive

root@wp-dev-blog2:/etc/varnish# curl -I localhost
HTTP/1.1 200 OK
Date: Thu, 07 Jul 2022 18:20:21 GMT
Content-Type: text/html
last-modified: Thu, 07 Jul 2022 04:30:44 GMT
vary: Accept-Encoding
etag: W/"62c66174-17c8"
```

```
Cache-Control: private
Age: 3
X-Cache: HIT
X-Cache-Hits: 1
Accept-Ranges: bytes
Content-Length: 6088
Connection: keep-alive
```

Varnish berhasil, ditunjukkan dengan dua header

```
X-Cache: HIT
X-Cache-Hits: 1
```

pada header `X-Cache`, jika nilainya `MISS` artinya belum di-cache oleh varnish, `HIT` artinya sudah ada cache.

From:
https://wiki.samsul.web.id/ - **Samsul Maarif**

Permanent link:
https://wiki.samsul.web.id/linux/Varnish.for.WordPress

Last update: **2022/12/28 16:34**