EBook Gratis

APRENDIZAJE varnish

Free unaffiliated eBook created from
Stack Overflow contributors.

#varnish

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: varnish

It is an unofficial and free varnish ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official varnish.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con el barniz

Observaciones

Esta sección proporciona una descripción general de qué es el barniz y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro del barniz y vincular a los temas relacionados. Dado que la Documentación para barniz es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Versión	Fecha de lanzamiento
5.1.2	2017-04-07
5.1.1	2017-03-16
5.0	2016-09-15
4.1.5	2016-02-09
4.0.4	2016-11-30
3.0.7	2015-03-23

Examples

Instalación o configuración

Las siguientes son instrucciones para configurar la última versión de Varnish en varias distribuciones de Linux.

CentOS 7

```
curl -s https://packagecloud.io/install/repositories/varnishcache/varnish5/script.rpm.sh |
sudo bash
```

Ubuntu

```
apt-get install apt-transport-https
curl https://repo.varnish-cache.org/GPG-key.txt | apt-key add -
echo "deb https://repo.varnish-cache.org/ubuntu/ trusty varnish-4.1" \
```

```
  >> /etc/apt/sources.list.d/varnish-cache.list
apt-get update
apt-get install varnish
```

Debian

```
apt-get install apt-transport-https
curl https://repo.varnish-cache.org/GPG-key.txt | apt-key add -
echo "deb https://repo.varnish-cache.org/debian/ jessie varnish-4.1"\
  >> /etc/apt/sources.list.d/varnish-cache.list
apt-get update
apt-get install varnish
```

Barniz VCL

Varnish controla y manipula las solicitudes HTTP utilizando Varnish Configuration Language (VCL). El siguiente fragmento de VCL elimina las cookies de las solicitudes entrantes al subdirectorio / images:

```
sub vcl_recv {
    if (req.url ~ "^/images") {
        unset req.http.cookie;
    }
}
```

Lea Empezando con el barniz en línea: https://riptutorial.com/es/varnish/topic/4705/empezando-con-el-barniz

Capítulo 2: Construyendo vmods

Introducción

No necesariamente tiene que compilar vmods si los binarios para ellos ya están disponibles para su plataforma. Tanto CentOS 6 como 7 pueden aprovechar las compilaciones COPR de Ingvar para instalar una colección de módulos adicionales de Varnish Software: https://copr.fedorainfracloud.org/coprs/ingvar/varnish51/

Examples

Compila e instala un vmod

La instalación de un vmod requiere una versión instalada de Varnish Cache, incluidos los archivos de desarrollo. Los requisitos se pueden encontrar en la documentación de Barniz.

El código fuente está construido con autotools:

```
sudo apt-get install libvarnishapi-dev || sudo yum install varnish-libs-devel
./bootstrap   # If running from git.
./configure
make
make check   # optional
sudo make install
```

Lea Construyendo vmods en línea: https://riptutorial.com/es/varnish/topic/9669/construyendo-vmods

Capítulo 3: Monitoreo de barniz

Introducción

Use `varnishstat` para monitorear las métricas numéricas de una instancia de Varnish actualmente en ejecución. Su ubicación diferirá según su instalación. La ejecución de `varnishstat -1` generará todas las métricas en un formato simple compatible con `grep` .

Otras utilidades están disponibles para ver el estado actual del barniz y el registro: `varnishtop` , `varnishlog` , etc.

Examples

Métricas del cliente - tráfico entrante

Las métricas del cliente cubren el tráfico entre el cliente y el caché de Barniz.

- sess_conn - Número acumulativo de conexiones.
- client_req - Número acumulado de solicitudes de clientes.
- sess_dropped: conexiones eliminadas debido a una cola completa.

Supervise `sess_conn` y `client_req` para realizar un seguimiento del volumen de tráfico: ¿está aumentando o disminuyendo? ¿Está aumentando? Los cambios repentinos pueden indicar problemas.

Monitorea `sess_dropped` para ver si el caché está eliminando alguna sesión. Si es así, es posible que necesite aumentar `thread_pool_max` .

```
varnishstat -1 | grep "sess_conn\|client_req \|sess_dropped"
MAIN.sess_conn              62449574         3.38 Sessions accepted
MAIN.client_req            184697229         9.99 Good client requests received
MAIN.sess_dropped                  0         0.00 Sessions dropped for thread
```

Rendimiento de caché

Quizás la métrica de rendimiento más importante es el hitrate.

Las rutas de barniz son las solicitudes entrantes como esta:

- Hash, una solicitud cacheable. Esto puede ser ya sea `hit` o `miss` en función del estado de la memoria caché.
- Hitpass, una solicitud no cacheable.

Un hash con un `miss` y un `hitpass` se `hitpass` en el servidor backend y se entregarán. Un hash con un `hit` enviará directamente desde el caché.

Métricas para monitorear:

- cache_hit - Número de hashes con un hit en el caché.
- cache_miss - Número de hashes con una falta en el caché.
- cache_hitpass - Número de hitpasses como los anteriores.

```
varnishstat -1 | grep "cache_hit \|cache_miss \|cache_hitpass"
MAIN.cache_hit              99032838         5.36 Cache hits
MAIN.cache_hitpass                 0         0.00 Cache hits for pass
MAIN.cache_miss             42484195         2.30 Cache misses
```

Calcula el hitrate real de esta manera:

```
cache_hit / (cache_hit + cache_miss)
```

En este ejemplo, el hitrato es 0.7 o 70%. Quieres mantener esto lo más alto posible. 70% es un número decente. Puede mejorar hitrate aumentando la memoria y personalizando su vcl. También monitorea grandes cambios en tu hitrate.

Monitoreo de objetos en caché

Usted supervisa los objetos almacenados en caché para ver con qué frecuencia expiran y si están "nuked".

- n_expirado - Número de objetos caducados.
- n_lru_nuked - Últimos objetos nuked utilizados recientemente. Número de objetos nuked (eliminados) de la caché debido a la falta de espacio.

```
varnishstat -1 | grep "n_expired\|n_lru_nuked"
MAIN.n_expired              42220159          .   Number of expired objects
MAIN.n_lru_nuked              264005          .   Number of LRU nuked objects
```

El que hay que ver aquí es `n_lru_nuked` , si la tasa está aumentando (la *tasa* , no solo el número), su caché está empujando los objetos cada vez más rápido debido a la falta de espacio. Necesitas aumentar el tamaño del caché.

El `n_expired` es más a su aplicación. Un tiempo de vida más largo disminuirá este número pero, por otro lado, no renovará los objetos tan a menudo. También el caché puede requerir más tamaño.

Monitoreo de hilos

Debe realizar un seguimiento de algunas métricas de hilos para ver su caché de barniz. ¿Se está quedando sin recursos del sistema operativo o está funcionando bien?

- hilos - Número de hilos en todos los grupos.
- threads_created - Número de hilos creados.
- threads_failed - Número de veces que Varnish no pudo crear un hilo.
- threads_limited - Número de veces que Varnish se vio forzado a no crear un hilo ya que estaba al máximo.

- thread_queue_len - Longitud de la cola actual. Número de solicitudes en espera de un hilo.
- sess_queued - Número de veces que no hubo subprocesos disponibles, por lo que una solicitud tuvo que ponerse en cola.

```
varnishstat -1 | grep "threads\|thread_queue_len\|sess_queued"
MAIN.threads                      100          .   Total number of threads
MAIN.threads_limited                1       0.00 Threads hit max
MAIN.threads_created             3715       0.00 Threads created
MAIN.threads_destroyed           3615       0.00 Threads destroyed
MAIN.threads_failed                 0       0.00 Thread creation failed
MAIN.thread_queue_len               0          .   Length of session queue
MAIN.sess_queued                 2505       0.00 Sessions queued for thread
```

Si `thread_queue_len` no es 0, significa que Varnish no tiene recursos y ha comenzado a poner en cola las solicitudes. Esto disminuirá el rendimiento de esas solicitudes. Necesitas investigar por qué.

Cuidado también con `threads_failed` . Si esto aumenta, significa que su servidor se ha quedado sin recursos de alguna manera. El aumento de los números en `threads_limited` significa que podría necesitar aumentar sus servidores `thread_pool_max` .

Monitoreando las métricas del backend.

Hay una serie de métricas que describen la comunicación entre Varnish y sus backends.

Las métricas más importantes aquí podrían ser estas:

- backend_busy - Número de estados http 5xx recibidos por un backend. Con VCL puede configurar Varnish para probar otro backend si esto sucede.
- backend_fail - Número de veces que Varnish no pudo conectarse al backend. Esto puede tener varias causas (no hay conexión TCP, mucho tiempo hasta el primer byte, mucho tiempo entre bytes). Si esto sucede tu backend no es saludable.
- backend_unhealthy - Número de veces que Varnish no pudo "hacer ping" al backend (no respondió con una respuesta HTTP 200).

```
varnishstat -1 | grep "backend_"
MAIN.backend_conn           86913481         4.70 Backend conn. success
MAIN.backend_unhealthy             0         0.00 Backend conn. not attempted
MAIN.backend_busy                  0         0.00 Backend conn. too many
MAIN.backend_fail                  7         0.00 Backend conn. failures
MAIN.backend_reuse                 0         0.00 Backend conn. reuses
MAIN.backend_toolate               0         0.00 Backend conn. was closed
MAIN.backend_recycle               0         0.00 Backend conn. recycles
MAIN.backend_retry                 0         0.00 Backend conn. retry
MAIN.backend_req            86961073         4.70 Backend requests made
```

Lea Monitoreo de barniz en línea: https://riptutorial.com/es/varnish/topic/9072/monitoreo-de-barniz

Capítulo 4: VCL incorporado

Introducción

El VCL incorporado contiene los procedimientos que se incluyen y ejecutan por *última vez* por Varnish.

Pueden complementar la VCL definida por el usuario al proporcionar la lógica adecuada para la mayoría de los sitios. Por ejemplo, omite la memoria caché para solicitudes POST y / o en presencia de cookies o encabezados de autorización.

Si parte de la lógica incorporada no es necesaria, un usuario puede agregar la llamada `return()` desde un procedimiento donde la lógica VCL incorporada no es deseable.

Examples

Barniz 3.0

```
/*-
 *
 * The default VCL code.
 *
 * NB! You do NOT need to copy & paste all of these functions into your
 * own vcl code, if you do not provide a definition of one of these
 * functions, the compiler will automatically fall back to the default
 * code from this file.
 *
 */

sub vcl_recv {
    if (req.restarts == 0) {
        if (req.http.x-forwarded-for) {
            set req.http.X-Forwarded-For =
                req.http.X-Forwarded-For + ", " + client.ip;
        } else {
            set req.http.X-Forwarded-For = client.ip;
        }
    }
    if (req.request != "GET" &&
      req.request != "HEAD" &&
      req.request != "PUT" &&
      req.request != "POST" &&
      req.request != "TRACE" &&
      req.request != "OPTIONS" &&
      req.request != "DELETE") {
        /* Non-RFC2616 or CONNECT which is weird. */
        return (pipe);
    }
    if (req.request != "GET" && req.request != "HEAD") {
        /* We only deal with GET and HEAD by default */
        return (pass);
    }
    if (req.http.Authorization || req.http.Cookie) {
```

```
        /* Not cacheable by default */
        return (pass);
    }
    return (lookup);
}

sub 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.
    return (pipe);
}

sub vcl_pass {
    return (pass);
}

sub vcl_hash {
    hash_data(req.url);
    if (req.http.host) {
        hash_data(req.http.host);
    } else {
        hash_data(server.ip);
    }
    return (hash);
}

sub vcl_hit {
    return (deliver);
}

sub vcl_miss {
    return (fetch);
}

sub vcl_fetch {
    if (beresp.ttl <= 0s ||
        beresp.http.Set-Cookie ||
        beresp.http.Vary == "*") {
                /*
                 * Mark as "Hit-For-Pass" for the next 2 minutes
                 */
                set beresp.ttl = 120 s;
                return (hit_for_pass);
    }
    return (deliver);
}

sub vcl_deliver {
    return (deliver);
}

sub vcl_error {
    set obj.http.Content-Type = "text/html; charset=utf-8";
    set obj.http.Retry-After = "5";
    synthetic {"
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>"} + obj.status + " " + obj.response + {"</title>
  </head>
  <body>
    <h1>Error "} + obj.status + " " + obj.response + {"</h1>
    <p>"} + obj.response + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + req.xid + {"</p>
    <hr>
    <p>Varnish cache server</p>
  </body>
</html>
"};
    return (deliver);
}

sub vcl_init {
        return (ok);
}

sub vcl_fini {
        return (ok);
}
```

Barniz 4.0

```
/*
 * The built-in (previously called default) VCL code.
 *
 * NB! You do NOT need to copy & paste all of these functions into your
 * own vcl code, if you do not provide a definition of one of these
 * functions, the compiler will automatically fall back to the default
 * code from this file.
 *
 * This code will be prefixed with a backend declaration built from the
 * -b argument.
 */

vcl 4.0;

######################################################################
# Client side

sub vcl_recv {
    if (req.method == "PRI") {
    /* We do not support SPDY or HTTP/2.0 */
    return (synth(405));
    }
    if (req.method != "GET" &&
      req.method != "HEAD" &&
      req.method != "PUT" &&
      req.method != "POST" &&
      req.method != "TRACE" &&
      req.method != "OPTIONS" &&
      req.method != "DELETE") {
        /* Non-RFC2616 or CONNECT which is weird. */
        return (pipe);
```

```
    }

    if (req.method != "GET" && req.method != "HEAD") {
        /* We only deal with GET and HEAD by default */
        return (pass);
    }
    if (req.http.Authorization || req.http.Cookie) {
        /* Not cacheable by default */
        return (pass);
    }
    return (hash);
}

sub vcl_pipe {
    # By default Connection: close is set on all piped requests, to stop
    # connection reuse from sending future requests directly to the
    # (potentially) wrong backend. If you do want this to happen, you can undo
    # it here.
    # unset bereq.http.connection;
    return (pipe);
}

sub vcl_pass {
    return (fetch);
}

sub vcl_hash {
    hash_data(req.url);
    if (req.http.host) {
        hash_data(req.http.host);
    } else {
        hash_data(server.ip);
    }
    return (lookup);
}

sub vcl_purge {
    return (synth(200, "Purged"));
}

sub vcl_hit {
    if (obj.ttl >= 0s) {
        // A pure unadultered hit, deliver it
        return (deliver);
    }
    if (obj.ttl + obj.grace > 0s) {
        // Object is in grace, deliver it
        // Automatically triggers a background fetch
        return (deliver);
    }
    // fetch & deliver once we get the result
    return (fetch);
}

sub vcl_miss {
    return (fetch);
}

sub vcl_deliver {
    return (deliver);
}
```

```
/*
 * We can come here "invisibly" with the following errors: 413, 417 & 503
 */
sub vcl_synth {
    set resp.http.Content-Type = "text/html; charset=utf-8";
    set resp.http.Retry-After = "5";
    synthetic( {"<!DOCTYPE html>
<html>
  <head>
    <title>"} + resp.status + " " + resp.reason + {"</title>
  </head>
  <body>
    <h1>Error "} + resp.status + " " + resp.reason + {"</h1>
    <p>"} + resp.reason + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + req.xid + {"</p>
    <hr>
    <p>Varnish cache server</p>
  </body>
</html>
"} );
    return (deliver);
}

#######################################################################
# Backend Fetch

sub vcl_backend_fetch {
    return (fetch);
}

sub vcl_backend_response {
    if (beresp.ttl <= 0s ||
      beresp.http.Set-Cookie ||
      beresp.http.Surrogate-control ~ "no-store" ||
      (!beresp.http.Surrogate-Control &&
        beresp.http.Cache-Control ~ "no-cache|no-store|private") ||
      beresp.http.Vary == "*") {
        /*
        * Mark as "Hit-For-Pass" for the next 2 minutes
        */
        set beresp.ttl = 120s;
        set beresp.uncacheable = true;
    }
    return (deliver);
}

sub vcl_backend_error {
    set beresp.http.Content-Type = "text/html; charset=utf-8";
    set beresp.http.Retry-After = "5";
    synthetic( {"<!DOCTYPE html>
<html>
  <head>
    <title>"} + beresp.status + " " + beresp.reason + {"</title>
  </head>
  <body>
    <h1>Error "} + beresp.status + " " + beresp.reason + {"</h1>
    <p>"} + beresp.reason + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + bereq.xid + {"</p>
```

```
    <hr>
    <p>Varnish cache server</p>
  </body>
</html>
"} );
    return (deliver);
}

#########################################################################
# Housekeeping

sub vcl_init {
    return (ok);
}

sub vcl_fini {
    return (ok);
}
```

Lea VCL incorporado en línea: https://riptutorial.com/es/varnish/topic/5001/vcl-incorporado

Creditos

S. No	Capítulos	Contributors
1	Empezando con el barniz	alejdg, Community, Daniel V.
2	Construyendo vmods	Daniel V.
3	Monitoreo de barniz	Jensd
4	VCL incorporado	Daniel V., fgsch, Redithion