eBook Gratuit

APPRENEZ varnish

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#varnish

Table des matières

À propos

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

Chapitre 1: Commencer avec le vernis

Remarques

Cette section fournit une vue d'ensemble de ce qu'est le vernis et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans le vernis, et établir un lien avec les sujets connexes. La documentation du vernis étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
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

Installation ou configuration

Vous trouverez ci-dessous des instructions pour configurer la dernière version de Varnish sur différentes distributions 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
```

Vernis VCL

Varnish contrôle et manipule les requêtes HTTP à l'aide de VCL (Varnish Configuration Language). L'extrait de VCL suivant supprime les cookies des demandes entrantes dans le sous-répertoire / images:

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

Lire Commencer avec le vernis en ligne: https://riptutorial.com/fr/varnish/topic/4705/commencer-avec-le-vernis

Chapitre 2: Construction de vmods

Introduction

Vous n'avez pas nécessairement besoin de compiler vmods si les fichiers binaires pour eux sont déjà disponibles pour votre plate-forme. CentOS 6 et 7 peuvent tous deux tirer parti des versions COPR d'Ingvar pour installer une collection de modules supplémentaires par Varnish Software: https://copr.fedorainfracloud.org/coprs/ingvar/varnish51/

Examples

Compiler et installer un vmod

L'installation d'un vmod nécessite une version installée de Varnish Cache, y compris les fichiers de développement. Les exigences peuvent être trouvées dans la documentation Varnish.

Le code source est construit avec des 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
```

Lire Construction de vmods en ligne: https://riptutorial.com/fr/varnish/topic/9669/construction-de-vmods

Chapitre 3: VCL intégré

Introduction

La VCL intégrée contient des procédures incluses et exécutées en *dernier* par Varnish.

Ils peuvent compléter la VCL définie par l'utilisateur en fournissant une logique adaptée à la majorité des sites. Par exemple, ignore le cache pour les requêtes POST et / ou en présence d'en-têtes de cookie ou d'autorisation.

Si une partie de la logique intégrée n'est pas nécessaire, un utilisateur peut ajouter un appel `return()` partir d'une procédure pour laquelle la logique VCL intégrée n'est pas souhaitable.

Examples

Vernis 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);
}
```

Vernis 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);
}
```

Lire VCL intégré en ligne: https://riptutorial.com/fr/varnish/topic/5001/vcl-integre

Chapitre 4: Vernis de surveillance

Introduction

Utilisez `varnishstat` pour surveiller les métriques numériques d'une instance Varnish en cours d'exécution. Son emplacement sera différent en fonction de votre installation. Exécuter `varnishstat -1` affichera toutes les métriques dans un simple format `grep` -able.

D'autres utilitaires sont disponibles pour surveiller l'état actuel du vernis et la journalisation: `varnishtop` , `varnishlog` etc.

Examples

Mesures client - trafic entrant

Les métriques client couvrent le trafic entre le client et le cache Varnish.

- sess_conn - Nombre cumulé de connexions.
- client_req - Nombre cumulé de demandes client.
- sess_dropped - Suppression de connexions en raison d'une file d'attente complète.

Surveillez `sess_conn` et `client_req` pour suivre le volume du trafic - augmente-t-il ou diminue-t-il, est-il en `client_req` de `client_req` , etc. Des changements soudains peuvent indiquer des problèmes.

Surveillez `sess_dropped` pour voir si le cache supprime des sessions. Si c'est le cas, vous devrez peut-être augmenter `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
```

Performances du cache

La mesure de performance la plus importante est peut-être le taux de réussite.

Le vernis achemine les requêtes entrantes comme ceci:

- Hash, une requête en cache. Cela peut être un `hit` ou un `miss` fonction de l'état du cache.
- Hitpass, une requête non cachable.

Un hachage avec un `miss` et un `hitpass` sera récupéré à partir du serveur de back - end et livré. Un hachage avec un `hit` sera envoyé directement depuis le cache.

Métriques à surveiller:

- cache_hit - Nombre de hachages avec un hit dans le cache.
- cache_miss - Nombre de hachages manquants dans le cache.
- cache_hitpass - Nombre de hitpass comme ci-dessus.

```
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
```

Calculez le hitrate actuel comme ceci:

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

Dans cet exemple, le taux de réussite est de 0,7 ou 70%. Vous voulez garder cela aussi haut que possible. 70% est un nombre décent. Vous pouvez améliorer le taux de réussite en augmentant la mémoire et en personnalisant votre vcl. Surveillez également les gros changements de votre taux de réussite.

Surveillance des objets en cache

Vous surveillez les objets mis en cache pour voir à quelle fréquence ils expirent et s'ils sont "nukés".

- n_expired - Nombre d'objets expirés.
- n_lru_nuked - Derniers objets nuked récemment utilisés. Nombre d'objets cachés (supprimés) du cache en raison d'un manque d'espace.

```
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
```

Le point à surveiller ici est `n_lru_nuked` , si le taux augmente (le *taux* , pas seulement le nombre), votre cache repousse les objets de plus en plus rapidement à cause du manque d’espace. Vous devez augmenter la taille du cache.

Le `n_expired` est plus à la hauteur de votre application. Un temps de vie plus long diminuera ce nombre mais ne renouvellera pas les objets aussi souvent. De plus, le cache peut nécessiter plus de taille.

Surveillance des threads

Vous devez garder une trace de certaines métriques de threads pour surveiller votre cache Varnish. Est-il à court de ressources du système d'exploitation ou fonctionne-t-il correctement?

- threads - Nombre de threads dans tous les pools.
- threads_created - Nombre de threads créés.
- threads_failed - Nombre de fois que le vernis n'a pas réussi à créer un thread.
- threads_limited - Nombre de fois où Varnish a été obligé de ne pas créer de thread depuis

qu'il a été maximisé.
- thread_queue_len - Longueur de la file d'attente actuelle. Nombre de demandes en attente d'un thread.
- sess_queued - Nombre de fois où aucun thread n'était disponible, une requête a donc dû être mise en file d'attente.

```
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` n'est pas 0, cela signifie que Varnish manque de ressources et a commencé à mettre les requêtes en attente. Cela diminuera la performance de ces demandes. Vous devez rechercher pourquoi.

Regardez aussi pour `threads_failed` . Si cela augmente, cela signifie que votre serveur manque de ressources. Augmenter les nombres dans `threads_limited` signifie que vous pourriez avoir besoin d'augmenter vos serveurs `thread_pool_max` .

Surveillance des métriques du backend

Il existe un certain nombre de mesures décrivant la communication entre Varnish et ses backends.

Les mesures les plus importantes ici pourraient être les suivantes:

- backend_busy - Nombre de statuts http 5xx reçus par un backend. Avec VCL, vous pouvez configurer Varnish pour essayer un autre backend si cela se produit.
- backend_fail - Nombre de fois que Varnish n'a pas pu se connecter au backend. Cela peut avoir plusieurs causes (pas de connexion TCP, long délai avant le premier octet, long délai entre les octets). Si cela se produit, votre backend n'est pas sain.
- backend_unhealthy - Nombre de fois où Varnish n'a pas pu "ping" le backend (il n'a pas répondu avec une réponse 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
```

Lire Vernis de surveillance en ligne: https://riptutorial.com/fr/varnish/topic/9072/vernis-de-surveillance

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec le vernis	alejdg, Community, Daniel V.
2	Construction de vmods	Daniel V.
3	VCL intégré	Daniel V., fgsch, Redithion
4	Vernis de surveillance	Jensd