Бесплатная электронная книга

учусь

varnish

Free unaffiliated eBook created from
Stack Overflow contributors.

#varnish

Около

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

глава 1: Начало работы с лаком

замечания

В этом разделе представлен обзор того, что такое лак, и почему разработчик может захотеть его использовать.

Следует также упомянуть любые крупные предметы в лаке и ссылки на связанные темы. Поскольку документация для лака новая, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Дата выхода
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

Установка или настройка

Ниже приведены инструкции по настройке последней версии Varnish на разных дистрибутивах 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
```

Varnish VCL

Larn контролирует и обрабатывает HTTP-запросы с использованием языка конфигурации Varnish (VCL). Следующий фрагмент VCL удаляет файл cookie из входящих запросов в / images подкаталог:

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

Прочитайте Начало работы с лаком онлайн: https://riptutorial.com/ru/varnish/topic/4705/начало-работы-с-лаком

глава 2: Встроенный VCL

Вступление

Встроенный VCL содержит процедуры, которые включены и выполняются *последним с* помощью лака.

Они могут дополнять пользовательский VCL, предоставляя логику, которая подходит для большинства сайтов. Например, пропускает кеш для запросов POST и / или в наличии заголовков файлов cookie или авторизации.

Если некоторая встроенная логика не нужна, пользователь может добавить вызов `return()` из процедуры, когда встроенная логика VCL нежелательна.

Examples

Лак 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);
}
```

Лак 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);
}
```

Прочитайте Встроенный VCL онлайн: https://riptutorial.com/ru/varnish/topic/5001/встроенный-vcl

глава 3: Мониторный лак

Вступление

Используйте `varnishstat` для контроля числовых показателей текущего исполняемого экземпляра Varnish. Местоположение будет отличаться в зависимости от вашей установки. Запуск `varnishstat -1` будет выводить все показатели в простом формате `grep` -able.

Другие утилиты доступны для просмотра текущего состояния лака и ведения журнала: `varnishtop` , `varnishlog` и т. Д.

Examples

Показатели клиента - входящий трафик

Метрики клиента охватывают трафик между клиентом и кэшем Varnish.

- sess_conn - Совокупное количество подключений.
- client_req - совокупное количество клиентских запросов.
- sess_dropped - Удаленные соединения из-за полной очереди.

Монитор `sess_conn` и `client_req` для отслеживания объема трафика - это увеличение или уменьшение, это пикирование и т. Д. Внезапные изменения могут указывать на проблемы.

Монитор `sess_dropped` чтобы посмотреть, `sess_dropped` ли кеш какие-либо сеансы. Если это так, вам может потребоваться увеличить `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
```

Производительность кеша

Возможно, самым важным показателем производительности является hitrate.

Лакирует маршруты входящих запросов:

- Хэш, кэш-запрос. Это может быть либо `hit` или `miss` в зависимости от состояния кэша.
- Hitpass, не кэшируемый запрос.

Хеш с `miss` и `hitpass` будет извлечен из бэкэнда сервера и доставлен. Хэш с `hit` будет доставлен непосредственно из кеша.

Показатели для мониторинга:

- cache_hit - количество хэшей с хитом в кеше.
- cache_miss - количество хэшей с пропуском в кеше.
- cache_hitpass - количество ударов, как указано выше.

```
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
```

Вычислите фактический показатель:

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

В этом примере показатель составляет 0,7 или 70%. Вы хотите сохранить это как можно выше. 70% - достойное число. Вы можете улучшить производительность, увеличив объем памяти и настроив vcl. Также отслеживайте большие изменения в вашем хитрете.

Мониторинг кэшированных объектов

Вы наблюдаете за кешированными объектами, чтобы узнать, как часто они истекают, и если они «уничтожены».

- n_expired - количество просроченных объектов.
- n_lru_nuked - Недавно использовались ядерные объекты. Количество объектов, удаленных из кэша из-за нехватки места.

```
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
```

Для просмотра здесь `n_lru_nuked` , если скорость увеличивается (*скорость* , а не только число), ваш кеш быстрее и быстрее выталкивает объекты из-за нехватки места. Вам нужно увеличить размер кеша.

`n_expired` больше подходит для вашего приложения. Более продолжительное время жизни уменьшит это число, но, с другой стороны, не возобновит объекты так часто. Кроме того, кеш может потребовать большего размера.

Мониторинг потоков

Вам нужно следить за некоторыми показателями потоков, чтобы посмотреть ваш лаковый кэш. У вас заканчиваются ресурсы ОС или он работает нормально.

- threads - количество потоков во всех пулах.
- threads_created - Количество созданных потоков.

- threads_failed - количество раз, когда лак не смог создать поток.
- threads_limited - Количество раз, когда лак был вынужден не создавать нить, так как он был максимальным.
- thread_queue_len - Текущая длина очереди. Количество запросов, ожидающих поток.
- sess_queued - количество раз, когда не было доступных потоков, поэтому запрос должен был быть поставлен в очередь.

```
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
```

Если `thread_queue_len` не равно 0, это означает, что у материала Varnish нет ресурсов и он начал отправлять запросы в очередь. Это снизит производительность этих запросов. Вам нужно выяснить причину.

Смотрите также для `threads_failed` . Если это будет увеличиваться, значит, ваш сервер как-то не работает. Увеличение числа в `threads_limited` означает, что вам может потребоваться увеличить количество серверов `thread_pool_max` .

Мониторинг базовых показателей

Существует ряд показателей, описывающих связь между лаком и его внутренними компонентами.

Наиболее важными показателями могут быть следующие:

- backend_busy - количество статусов http 5xx, полученных бэкэнд. С VCL вы можете настроить Varnish, чтобы попробовать другой бэкэнд, если это произойдет.
- backend_fail - Количество раз, когда лак не мог подключиться к серверу. Это может иметь несколько причин (отсутствие TCP-соединения, длительное время до первого байта, длительное время между байтами). Если это произойдет, ваш бэкэнд не здоров.
- backend_unhealthy - количество раз, когда лак не мог «пинговать» бэкэнд (он не ответил ответом 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
```

Прочитайте Мониторный лак онлайн: https://riptutorial.com/ru/varnish/topic/9072/мониторный-лак

глава 4: Строительство vmods

Вступление

Вам необязательно компилировать vmods, если для них уже доступны двоичные файлы для вашей платформы. Оба CentOS 6 и 7 могут использовать сборки COPR от Ingvar для установки набора дополнительных модулей с помощью программного обеспечения Varnish: https://copr.fedorainfracloud.org/coprs/ingvar/varnish51/

Examples

Скомпилируйте и установите vmod

Для установки vmod требуется установленная версия Varnish Cache, включая файлы разработки. Требования можно найти в документации по лакированию.

Исходный код построен с помощью 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
```

Прочитайте Строительство vmods онлайн: https://riptutorial.com/ru/varnish/topic/9669/строительство-vmods

кредиты

S. No	Главы	Contributors
1	Начало работы с лаком	alejdg, Community, Daniel V.
2	Встроенный VCL	Daniel V., fgsch, Redithion
3	Мониторный лак	Jensd
4	Строительство vmods	Daniel V.