Discourse PURGE cache method on content changes

Lee_Ars **Nov '19**

> maltfield:
>
> Pinging @Lee_Ars – how did you get varnish cache invalidation working with Discourse?

I sidestepped the issue altogether by having Varnish only cache static assets. In `sub vcl_recv` I have the following:

```
# Cache only the static assets in Discourse's "assets" dir and pass ever
if (req.http.host ~"discourse.bigdinosaur.org") {
    if (!(req.url ~ "(^/uploads/|^/assets/|^/user_avatar/)" )) {
        return (pass);
    }
}
```

As @codinghorror & others bring up, trying to cache full Discourse pages didn't seem like the correct way to go forward given Discourse's heavy use of dynamic content everywhere. I figured rather than trying to bang my head against a wall, I'd just cache images and call it done.

Ad

Michael Altfield maltfield **Dec '19**

> riking:
>
> nginx will directly serve uploads (sometimes), the Rails app passes off instructions to nginx to serve brotli versions of static content, the anon cache mentioned above, plus the Rails cache at the bottom which can result in "partially cached" requests.

Can someone please link me to the documentation that enumerates all these components, what each does, and how they can be configured?

> maltfield:
>
> Does Discourse invalidate its anonymous cache when there's updates to the backend? Or does `ANON_CACHE_DURATION` define the maximum amount of time for which content will be served to clients stale?
>
> What would be the consequences of setting ANON_CACHE_DURATION to 24 hours?

re-iterating this question from above

Michael Altfield maltfield 8h

FYI, here’s the varnish config I’ve created so far.

A few notes about this:

1. At the time of writing, this is for a test site that’s not receiving production traffic and may require more tuning.
2. Discourse doesn’t support PURGE, so we only cache for 1-5 minutes. If Discourse actually implemented support to PURGE content in a configurable set of cache servers, then we could increase this to ~24 hours like the rest of our sites. Until then, a low TTL may work well enough to mitigate against thundering herd “hug-of-death” crashes while having a worst-case delay of 1-5 minute stale content
3. POST content always bypasses the cache (ie: calls to /message-bus/)
4. Logged-in user requests (with cookie headers) bypass the cache
5. Static-assets (by an allow-list of file extensions) are are almost always cached, even when the user is logged-in.

```
################################################################################
```

```
# File:    discourse.opensourceecology.org.vcl
# Version: 0.3
# Purpose: Confg file for ose's discourse site
# Author:  Michael Altfield <michael@opensourceecology.org>
# Created: 2020-03-23
# Updated: 2020-05-18
################################################################################

vcl 4.0;

##########################
# VHOST-SPECIFIC BACKEND #
##########################

backend discourse_opensourceecology_org {
  .host = "127.0.0.1";
  .port = "8020";
}

sub vcl_recv {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    set req.backend_hint = discourse_opensourceecology_org;

    # Serve objects up to 2 minutes past their expiry if the backend
    # is slow to respond.
    #set req.grace = 5m;

    # append X-Forwarded-For for the backend
    # (note: this is already done by default in varnish >= v4.0)
    #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;
    #}

    # This uses the ACL action called "purge". Basically if a request to
    # PURGE the cache comes from anywhere other than localhost, ignore it.
    if (req.method == "PURGE") {
      if (!client.ip ~ purge) {
        return (synth(405, "Not allowed."));
      } else {
        return (purge);
      }
    }
```

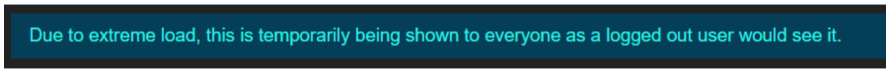
```
# Pass any requests that Varnish does not understand straight to the bac
if (
 req.method != "GET" && req.method != "HEAD" &&
 req.method != "PUT" && req.method != "POST" &&
 req.method != "TRACE" && req.method != "OPTIONS" &&
 req.method != "DELETE"
) {
  return (pipe);
} /* Non-RFC2616 or CONNECT which is weird. */

# Pass anything other than GET and HEAD directly.
if (req.method != "GET" && req.method != "HEAD") {
  return (pass);
}      /* We only deal with GET and HEAD by default */

# cache static content, even if a user is logged-in (but strip cookies)
if (req.method ~ "^(GET|HEAD)$" && req.url ~ "\.(jpg|jpeg|gif|png|ico|cs

  # if you use a subdomain for admin section, do not cache it
  #if (req.http.host ~ "admin.yourdomain.com") {
  #    set req.http.X-VC-Cacheable = "NO:Admin domain";
  #    return(pass);
  #}
  # enable this if you want
  #if (req.url ~ "debug") {
  #    set req.http.X-VC-Debug = "true";
  #}
  # enable this if you need it
  #if (req.url ~ "nocache") {
  #    set req.http.X-VC-Cacheable = "NO:Not cacheable, nocache in URL";
  #    return(pass);
  #}

  set req.url = regsub(req.url, "\?.*$", "");

  # unset cookie only if no http auth
  if (!req.http.Authorization) {
      unset req.http.Cookie;
  }

  return(hash);

}

# Pass requests from logged-in users directly.
# Only detect cookies with "session" and "Token" in file name, otherwise
```

```
    if (req.http.Authorization || req.http.Cookie ~ "session" || req.http.Co
      return (pass);
    } /* Not cacheable by default */

    # Pass any requests with the "If-None-Match" header directly.
    if (req.http.If-None-Match) {
      return (pass);
    }

    if (req.http.Cache-Control ~ "no-cache") {
      # swap these to respect client "no-cache" requests
      #ban(req.url);
      unset req.http.Cache-Control;
    }

    return (hash);

  }

}

sub vcl_hash {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    # TODO

  }
}

sub vcl_backend_response {

  if ( beresp.backend.name == "discourse_opensourceecology_org" ){

    # Avoid caching error responses
    if ( beresp.status != 200 && beresp.status != 203 && beresp.status != 30
      set beresp.ttl   = 0s;
      set beresp.grace = 15s;
      return (deliver);
    }

    if (!beresp.ttl > 0s) {
      set beresp.uncacheable = true;
      return (deliver);
    }

    if (beresp.http.Set-Cookie) {
      set beresp.uncacheable = true;
      return (deliver);
    }


    # discourse wasn't designed to play well with reverse proxy caching; ign
    # its ignorant no-cache headers (instead we just use a very short ttl)
    if (beresp.http.Cache-Control ~ "(private|no-cache|no-store)") {
      unset beresp.http.Cache-Control;
    }

    if (beresp.http.Authorization && !beresp.http.Cache-Control ~ "public")
      set beresp.uncacheable = true;
      return (deliver);
    }

    # always cache for 1-5 minutes with Discourse; we shouldn't set this to
    # because Discourse doesn't support PURGE. For more info, see:
    #  * https://meta.discourse.org/t/discourse-purge-cache-method-on-conten
    set beresp.ttl = 1m;
    set beresp.grace = 5m;
    return (deliver);

  }

}

sub vcl_synth {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    # TODO

  }

}

sub vcl_pipe {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    # 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 req.http.connection = "close";

    # This is otherwise not necessary if you do not do any request rewriting
    #set req.http.connection = "close";

  }
}

sub vcl_hit {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    # this is left-over from copying this config from the wiki's varnish con
    # but it won't actually be used until Discourse implements PURGE

    if (req.method == "PURGE") {
      ban(req.url);
      return (synth(200, "Purged"));
    }

    if (!obj.ttl > 0s) {
      return (pass);
    }

  }
}

sub vcl_miss {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    if (req.method == "PURGE")  {
      return (synth(200, "Not in cache"));
    }

  }
}

sub vcl_deliver {

  if ( req.http.host == "discourse.opensourceecology.org" ){

    # TODO

  }
}
```

Kane York riking team 2h

> maltfield:
>
> Until then, a low TTL may work well enough to mitigate against thundering herd “hug-of-death” crashes while having a worst-case delay of 1-5 minute stale content

Discourse already has “hug-of-death” load shedding built in - if requests start to pile up for a particular path, everyone will get the anonymous view, which gets cached:


Due to extreme load, this is temporarily being shown to everyone as a logged out user would see it.

Note that this protection **is applied to logged-in users as well**, which your Varnish config does not. Aggressive application-level caching is also already applied to anonymous requests at all times.

Michael Altfield maltfield 1h

Can you please point me to the documentation that describes this caching system and describes how it can be tuned?

Jeff Atwood codinghorror co-founder 1h

Sure, it’s the source code at https://github.com/discourse/discourse – have a read 📖

Michael Altfield maltfield **33m**

Wow.

Serious question: Does the discourse team have a policy against writing documentation? Does the Discourse project have any documentation-related policies at all?

Jeff Atwood codinghorror co-founder **32m**

We do have a policy against people being rude to us. Would you like to explore that policy now?

Kane York riking team **31m**

I've never experienced it needing to be tuned, except for when people didn't realize their servers were capping out. If you run into any actual issues with performance, instead of imaginary issues, come here and we can try to work it out.

Michael Altfield maltfield **31m**

@Jeff I appreciate the Discourse project and your open source contributions. We're using Discourse for an open-source project. We all love and appreciate the work people do for the community.

But code alone is not a substitute for documentation.

I'm not trying to be rude. It's a serious question: do you have policies on documentation? I keep hitting this wall, and I'd really appreciate an answer.

Jeff Atwood codinghorror co-founder **30m**

You're asking very narrow questions that don't come up often, and in ways that imagine speculative outcomes rather than based on actual experience and data. We don't tend to spend time on that, as historically it is not a good use of our engineering budget.

If you'd like to pay us to spend time on it, please sign up for a hosting plan at **https://discourse.org/buy** – I recommend enterprise level hosting if you want the level of support you're asking for here.

Michael Altfield maltfield **28m**

Do you have any policies on documentation?

We're a small nonprofit and have a $0 budget for this. All our developers are volunteers. Our project is a labour of love with a goal of providing open-source to the masses–to lower barriers and empower folks. And we do place a heavy emphasis on documentation; it's a critical component for open-source collaboration.

Do you have any policies on documentation?

Jeff Atwood codinghorror co-founder **27m**

I'll just repeat myself and emphasize the relevant sentences

> codinghorror:
>
> You're asking very narrow questions that don't come up often, and in ways that imagine speculative outcomes rather than based on actual experience and data. We don't tend to spend time on that, as historically it is not a good use of our engineering budget.

Sorry if you disagree with my assessment. But we run the project, not you. If that's a problem, perhaps pick a different free open source solution that is more to your liking.

Michael Altfield maltfield **1m**

Uh, I just noticed this:

> This topic will be automatically deleted in 14 days.

Please tell me this means this thread will become read-only and not deleted.

I’ve gone through a lot of effort using this thread as a means to provide documentation to other users, and I want to ensure that it won’t be deleted…