eZ Publish and Varnish

Web Summer Camp 2017

About me

- Hrvoje Knežević
    - Working at Netgen since 2012
    - Senior Symfony / eZ Publish developer, project tech lead
- https://twitter.com/hknezevic
- https://www.linkedin.com/in/hknezevic/
- https://gitlab.com/hknezevic
- https://github.com/hknezevic
- email: hrvoje.k@netgen.hr

…before we start…

- $ cd /var/www/html/summercamp/ezhttpcaching
- $ git pull
- $ ./prepare.sh

Varnish

- HTTP proxy
- HTTP accelerator for content-heavy dynamic web sites
- stores data in virtual memory, streams log information in shared memory
- heavily threaded

VCL

- *Varnish Configuration Language*
- domain-specific language, simple syntax similar to C (operators) or script languages (regex)
- translated to C, compiled as a shared object
- subroutines, ACL

Functions, actions and variables

- https://book.varnish-software.com/4.0/chapters/VCL_Basics.html
- https://varnish-cache.org/docs/4.0/reference/vcl.html
- https://varnish-cache.org/docs/4.0/users-guide/vcl-actions.html
- https://www.varnish-cache.org/docs/trunk/users-guide/vcl-built-in-subs.html

Functions, actions and variables

- vcl_recv, vcl_fetch
- pass, deliver, pipe, hash
- **req**, bereq, **resp**, beresp, obj(.ttl/.grace, …)

Varnish configuration file

- https://github.com/ezsystems/ezplatform/blob/master/doc/varnish/vcl/varnish4.vcl

X-User-Hash

- User context (additional Vary header)
- In the context of eZ Platform, X-User-Hash is generated based on the roles defined for user and user groups
- Anonymous user has hard-coded hash in VCL - prevents additional backend requests to retrieve user hash

```
$ curl -I -H "Accept: application/vnd.fos.user-context-hash" http://
ezhttpcaching.websc/_fos_user_context_hash
```

Tools

- **varnishlog**
  - displays formatted log stream from shared memory
  - https://www.varnish-cache.org/docs/5.0/reference/varnishlog.html
  - https://www.varnish-cache.org/trac/wiki/VarnishlogExamples

Tools

- **varnishadm**
  - CLI tool for accessing Varnish admin console

```
varnishadm "ban req.http.host ~ ezhttpcaching.websc"

varnishadm "ban req.http.host == ezhttpcaching.websc && req.url == /Projects"

varnishadm "ban req.http.host ~ ezhttpcaching.websc && req.url ~ .css$"

varnishadm "ban obj.http.X-ContentType-Identifier == blog_post"
```

Varnish and eZ

- https://doc.ez.no/display/EZP/Using+Varnish
- Varnish is used as an HTTP cache
- comparison to eZ Publish legacy
    - “replaces” legacy view cache
    - ESI fragments “replace” legacy block cache

Configuration

- apache/htaccess :

```
SetEnv SYMFONY_ENV prod

SetEnv SYMFONY_HTTP_CACHE 0

SetEnv SYMFONY_TRUSTED_PROXIES "127.0.0.1"
```

- nginx:

```
fastcgi_param SYMFONY_ENV prod;

fastcgi_param SYMFONY_HTTP_CACHE 0;

fastcgi_param SYMFONY_TRUSTED_PROXIES "127.0.0.1";
```

Configuration

- ezplatform.yml:

```
ezpublish:
    http_cache:
        purge_type: http
    system:
        site_group:
            http_cache:
                purge_servers: [http://127.0.0.1]
        default:
            content:
                view_cache: true
                ttl_cache: true
                default_ttl: 3600
```

Edge-Side Includes (ESI)

- small markup language for edge level dynamic web content assembly
- HTML statements over HTTP for including fragments of web pages in other web pages
- <esi:include src=…/>

Debugging ESI

```
$ php app/console cache:clear --no-debug

$ curl -H "Surrogate-Capability: abc=ESI/1.0" http://ezhttpcaching.websc
```

Edge-side includes

- eZ content views:

```
{{ render_esi( controller(
        "ez_content:viewAction", {"contentId": 123, "viewType": "line"}
)) }}
```

- Custom Symfony controllers:

```
{{ render_esi(controller("AppBundle:Parts:getLatestBlogPosts")) }}
```

Edge-side includes

- Symfony\Component\HttpFoundation\Response
    - setPublic()
    - setPrivate()
    - setMaxAge()
    - setSharedMaxAge()
    - setVary()
    - headers

Tagging view responses with custom headers

- eZ sets X-Location-Id header by default on every view_content response
- https://github.com/ezsystems/ezpublish-kernel/blob/master/eZ/Bundle/EzPublishCoreBundle/EventListener/CacheViewResponseListener.php
- we can implement our own response listeners for tagging content with additional headers
    - tag: kernel.event_subscriber

Smart view cache clearing

- provides possibility to clear cache for locations or content that is in relation with the content being currently cleared
    - tag: ezpublish.http_cache.event_subscriber
- https://github.com/ezsystems/ezpublish-kernel/tree/master/eZ/Publish/Core/MVC/Symfony/Cache/Http/EventListener
- extend list of location IDs which will be invalidated

Clearing cache by custom header

- We can utilise the fos_http_cache.cache_manager to manually invalidate HTTP cache
    - https://github.com/FriendsOfSymfony/FOSHttpCacheBundle/blob/master/src/CacheManager.php
    - https://github.com/FriendsOfSymfony/FOSHttpCache/blob/master/src/CacheInvalidator.php

Legacy tip

- If you are using the eZ Platform with legacy support, make use of the legacy smart view cache!
- Legacy view cache clearing triggers invalidation of the HTTP cache as well

Pro tip for the multi-site setups

- Varnish tags all of the responses with the X-Host header by default
- You can override the default FOSPurgeClient and apply additional headers on the BAN operations
    - https://github.com/ezsystems/ezpublish-kernel/blob/master/eZ/Publish/Core/MVC/Symfony/Cache/Http/FOSPurgeClient.php

What’s next?

- platform-http-cache
  - https://github.com/ezsystems/ezplatform-http-cache
  - Experimental HTTP cache handling for eZ Platform.
  - Utilizes xkey module for Varnish