novactive

eZ Publish Summercamp 2012

eZ Summer Camp 2012

eZ Publish Varnish Integration

How to optimize your eZ Publish with Varnish Reverse Proxy

Novactive

About us

NOVACTIVE, in a nutshell

Nicolas GENTY
Director

Bruno BASTET
VP of WW Sales

Damien LHOUILLIER
CTO

TEAM (75 employees)

- Top management : 3
- Sales : 9
- Consulting & Projet management : 19
- Design : 5
- Engineering and Development : 38

3 locations : Paris, Montréal et Tunis

Key Figures

- Creation date: 1996
- Net Worth (Dec10): $486K
- Sales (2011): aprox: $5M
- Workforce 2012: 75

PARIS OFFICE

42/44 rue du Paradis
75010 Paris
France

MONTREAL OFFICE

360 rue Saint Jacques
Ouest - Suite 1805
Montréal, QC H2Y 1P5
Canada

MAIN CUSTOMERS

oseo, SwissLife, BPCE, cea, Hammerson, EMI Music France, Centrale Paris, La Poste, Figaro Medias affluent people, la Nouvelle République.fr, AIRFRANCE KLM

WHAT WE DO?

Consulting & Expertise: Consulting, support design and implementation for Web & Mobile, Design, Online Marketing

Web and mobile development: Web: Expertise in cutting-edge Open Source technologies, mobile iPhone and Android Development

PARTNERSHIP & EXPERTISE

- **Platinum eZPublish Partner**
- **Varnish** partner
- **Emailvision** partner
- **certified ISO 9001 v. 2008**
  In 2009, 2010 and 2011
- Expertise **on mobile (Web and native Apps)**

About me

Engineering and Innovation Director @ Novactive

- Quick summary
  - Since 2003 at Novactive
  - Since 2005 on eZ Publish
  - Since 2010 on iOS
  - Since 2010 with Varnish

- Twitter : Plopix
- Blog : http://blog.plopix.net
- Mail : seb@novactive.com

Summary

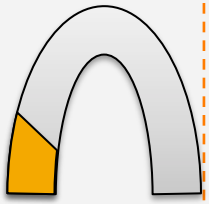
- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Summary

- Novactive
- ***Reverse Proxy***
- ***Varnish eZ Publish Architecture***
- ***Varnish Basics***
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Reminder of basics

- Architecture
- Varnish
- VCL
- Varnish Flow
- MISS / HIT definition
- Installation and Configuration

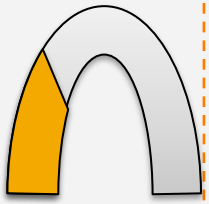


Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- ***Varnish Advanced #1***
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

VCL tips and tricks

- Debug
- ACL
- C code
- Implementations

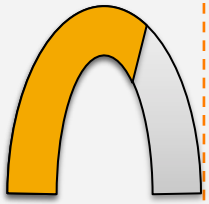


Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ***ESI Basics***
- ***ESI with eZ Publish***
- ***Recursive ESI***
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Edge Side Include

- Defintion
- Tips and tricks
- ESI content view
- Recursive usage
- Implementations



Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- ***Varnish purge on eZ Publish publication***
- Varnish Advanced #2

Instant publishing

- Tricks
- Unleash the TTL
- Varnish purge
- Implementation

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- ***Varnish Advanced #2***

Varnish daemon

- Tips and Tricks
- VCL advanced
- Performance optimization
- Troubeshoot your instance
- Implementation

Summary

- Novactive
- ***Reverse Proxy***
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Reverse Proxy

General

Reverse Proxy features

Features

- Handles a HTTP Cache
- Handles ESI (Edge Side Include)

Advantages

- Optimize the usage of your webserver
  - Therefore of your database
- Allows us to handle high peaks of traffic

Disadvantages

- Works on a short TTL (Time To Live)
- Delaying the publication with 2xTTL time
- Induced time lag of contents
- The permanent dilemma:
  - The greater the TTL is, the older the information could be
  - The more you reduce the TTL, the less you handle high traffic

Main Open Source Reverse Proxy in the world

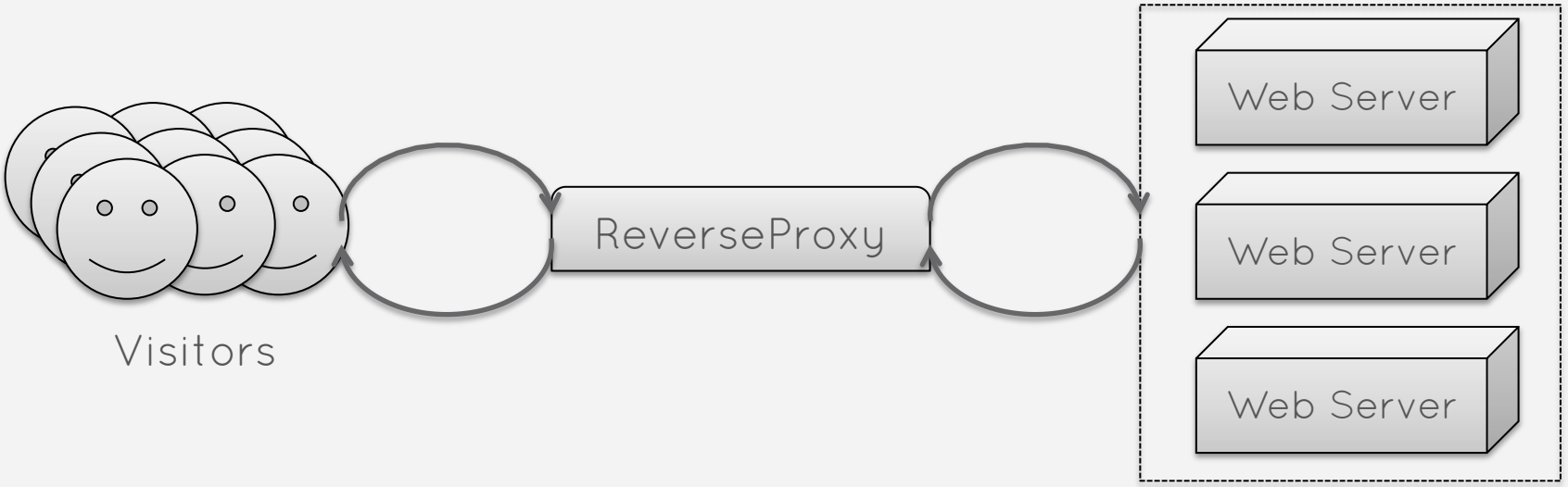
- Squid
- Apache
- Lighthttpd
- Nginx
- ***Varnish***

Architecture

- Clients
  - Resources who issues the request
- Backends
  - The different webservers behind Varnish
- Backend storage
  - The cache storage method
- Configuration
  - Set of rules to manage application

Reverse Proxy architecture

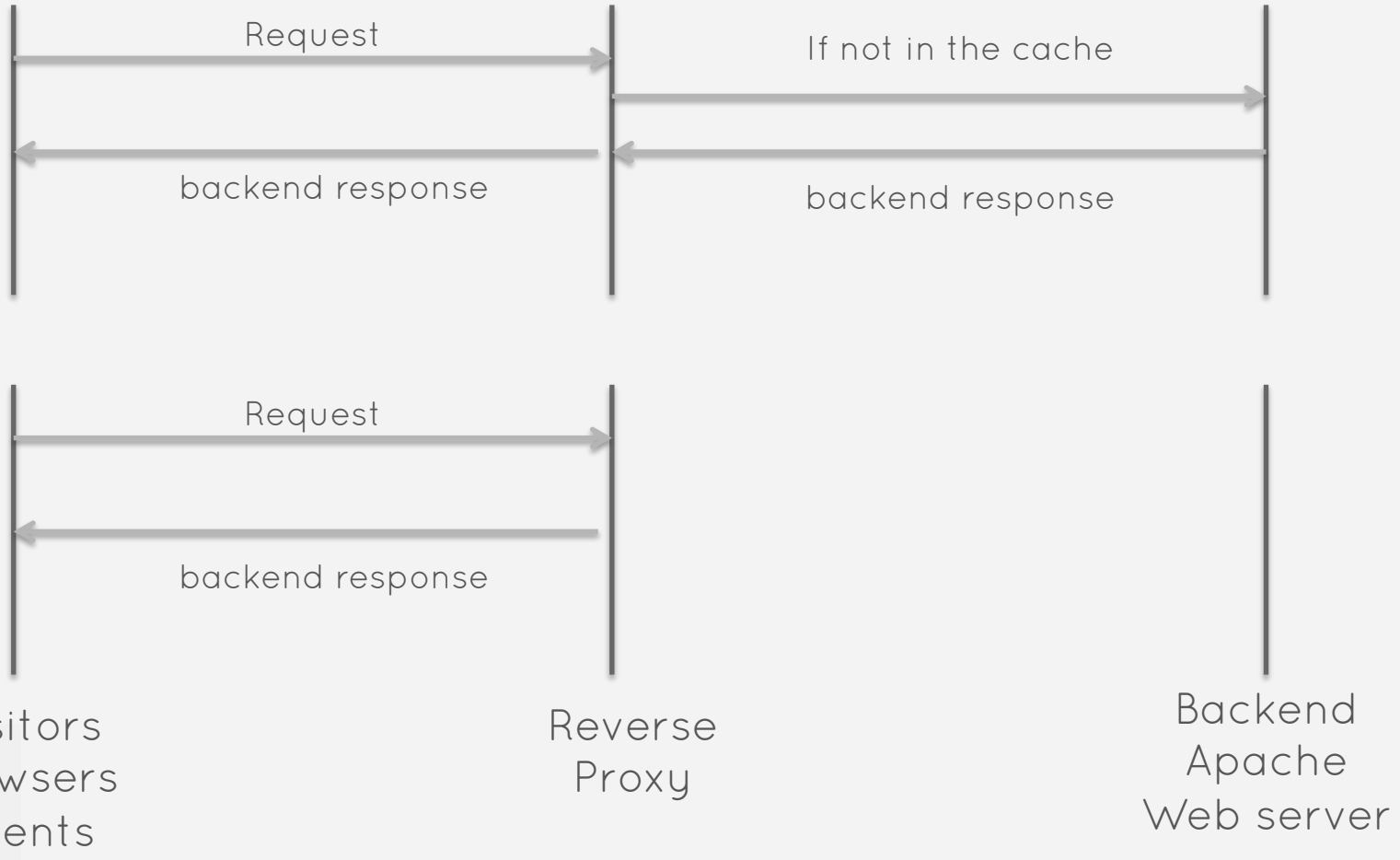
Basic platform integration



- You can see a reverse proxy as a shield of your web server farm.

Reverse Proxy architecture

Generic diagram (MISS and HIT)



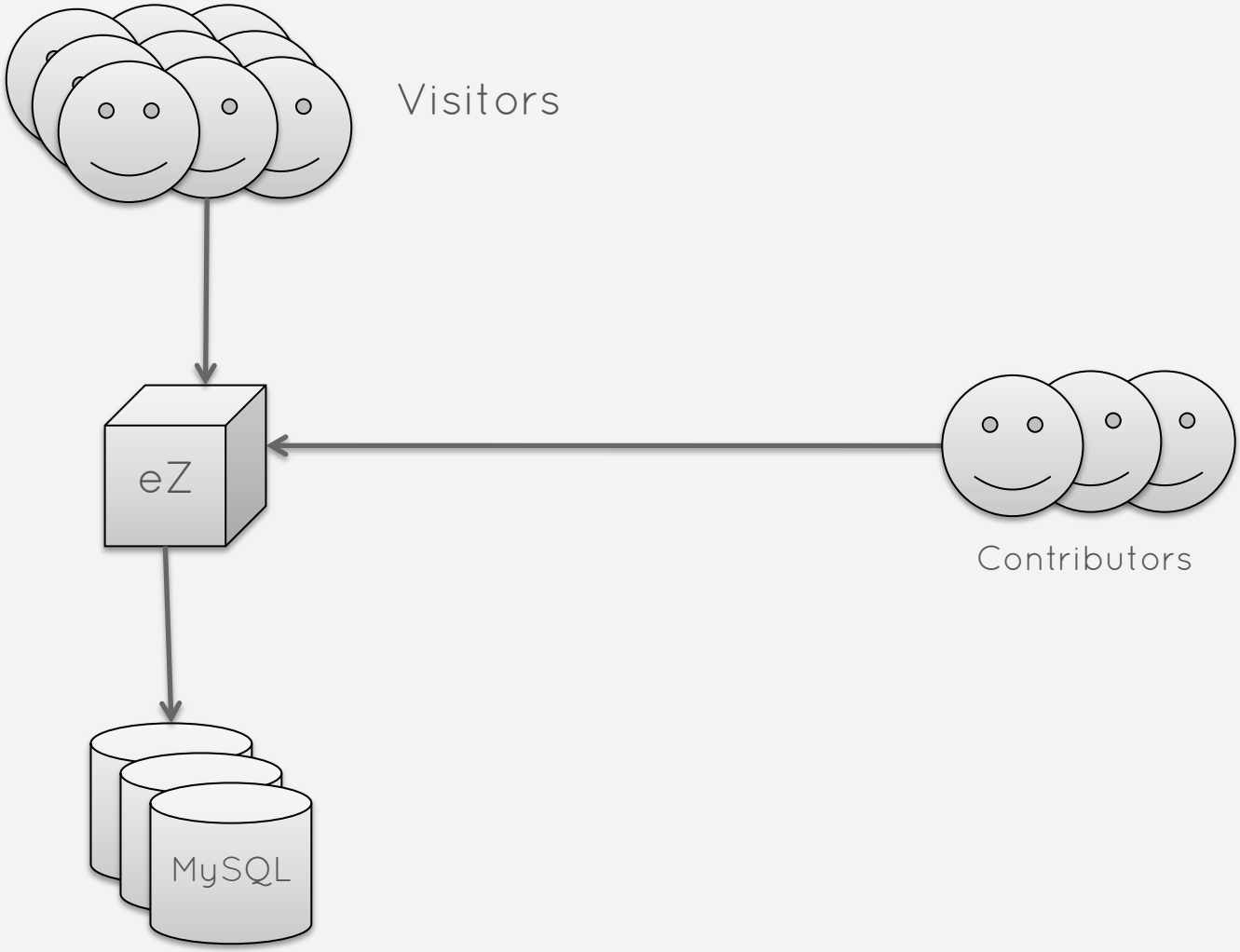
Summary

- Novactive
- Reverse Proxy
- ***Varnish eZ Publish Architecture***
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

eZ Publish without Varnish

Classic eZ Publish Architecture

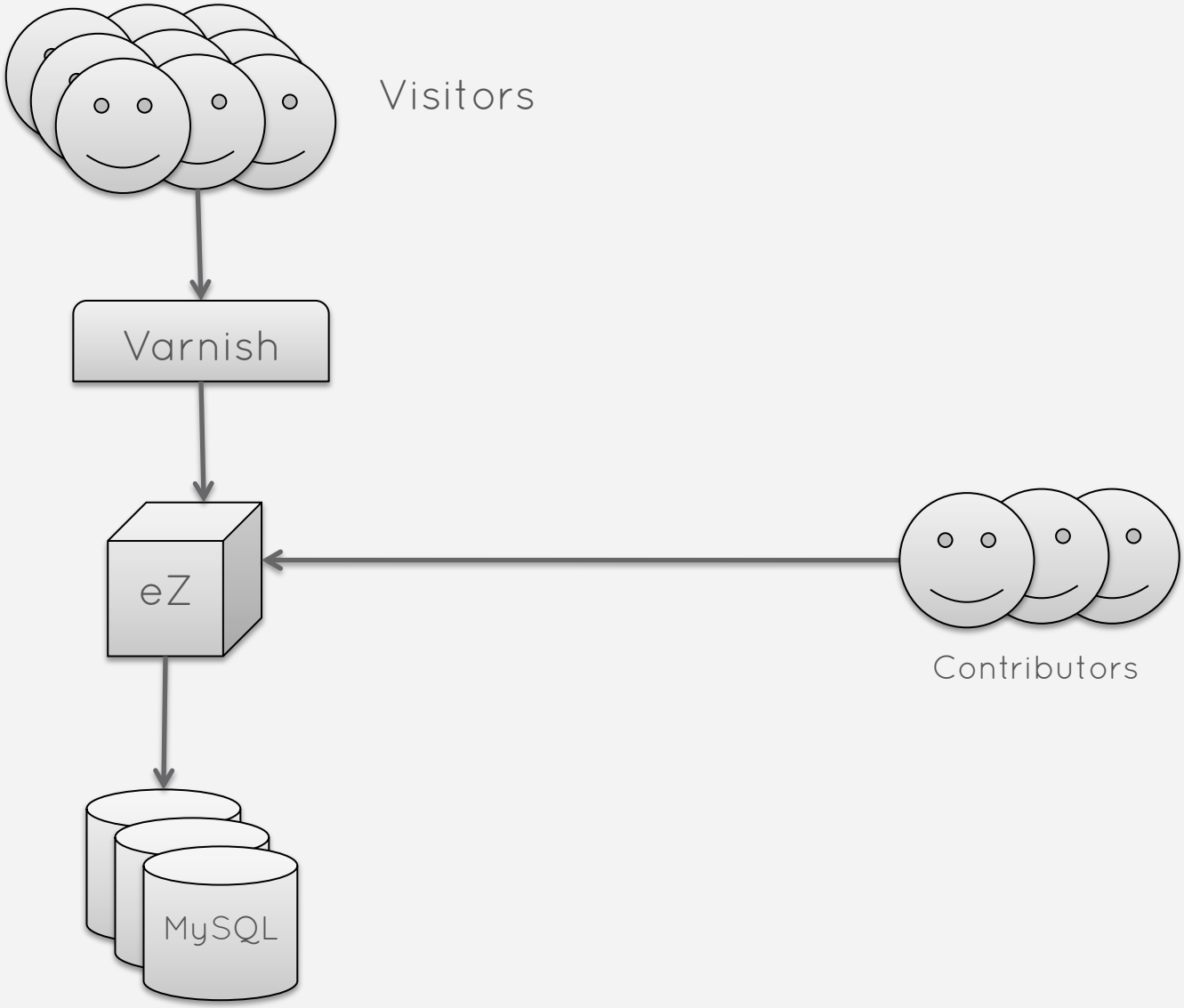
Classic architecture



eZ Publish with Varnish

Classic eZ Publish with Varnish

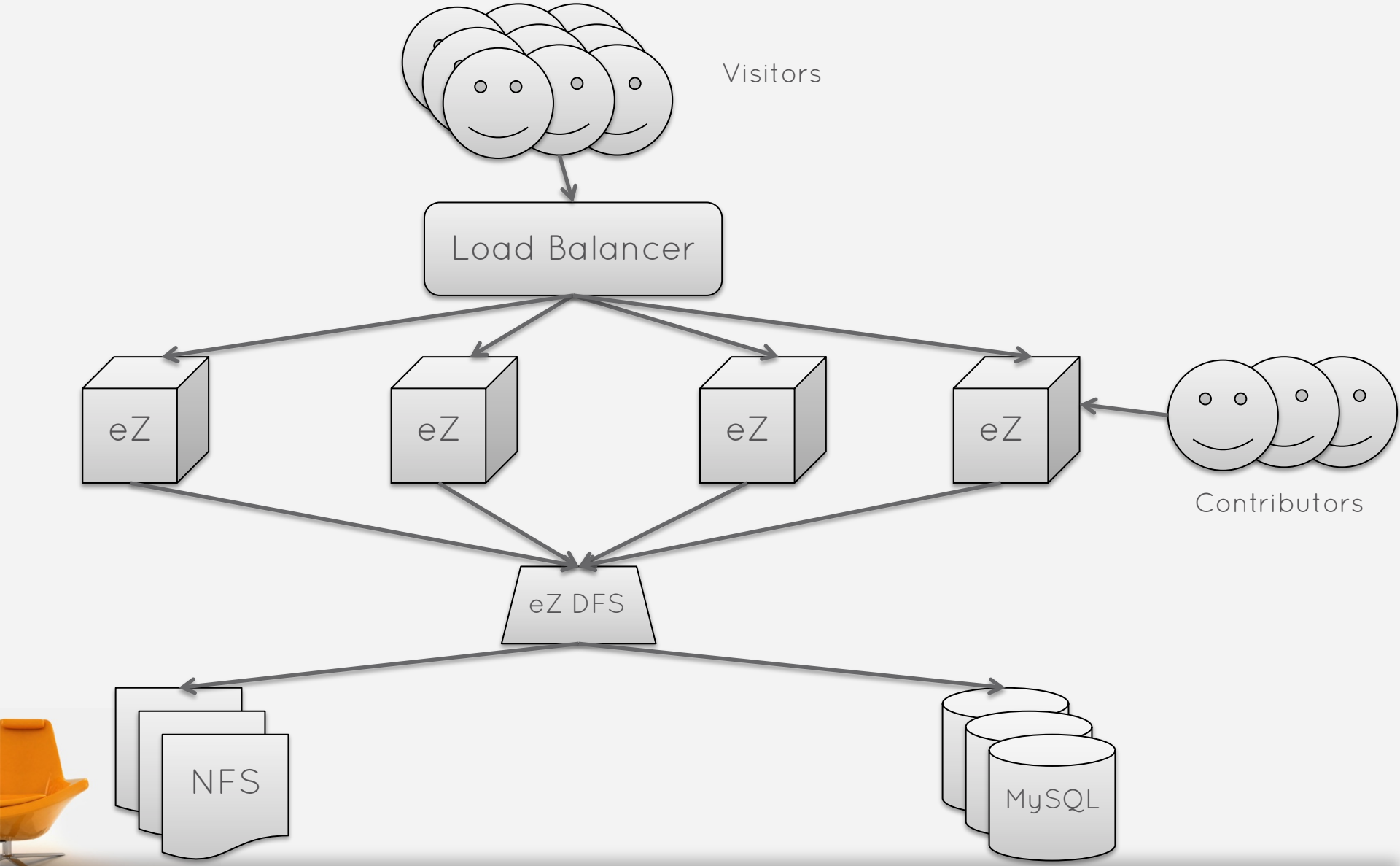
Classic Architecture + Varnish



eZ Publish Cluster without Varnish

eZ DFS Classic eZ Publish Architecture

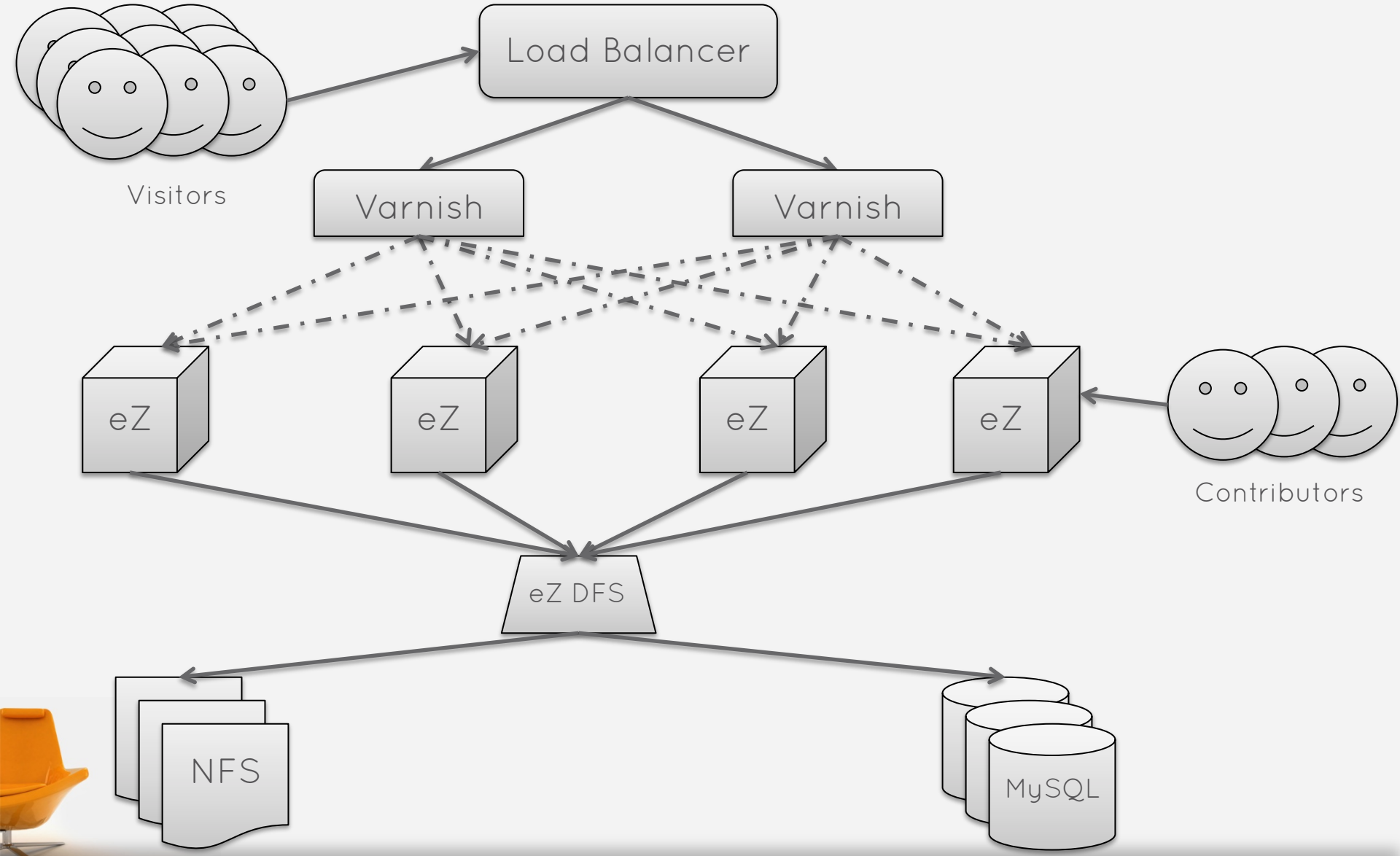
eZ DFS Classic architecture



eZ Publish Cluster with Varnish

eZ DFS eZ Publish with Varnish

eZ DFS Architecture + Varnish



Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- ***Varnish Basics***
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Varnish basics
Configuration

Installation

Configuration in service mode

- /etc/default/varnish
  - Service global configuration

- /etc/varnish/yourconfiguration.vcl
  - All your rules to handle HTTP cache in your application

Global configuration

- Socket port and socket admin port
- Memory
- Process limitation
- Default TTL
- ***VCL file path***

VCL : Varnish Configuratin Language

- Syntax from C and Perl
- No loops, no variables

=> It describes the caching policy !

Syntax

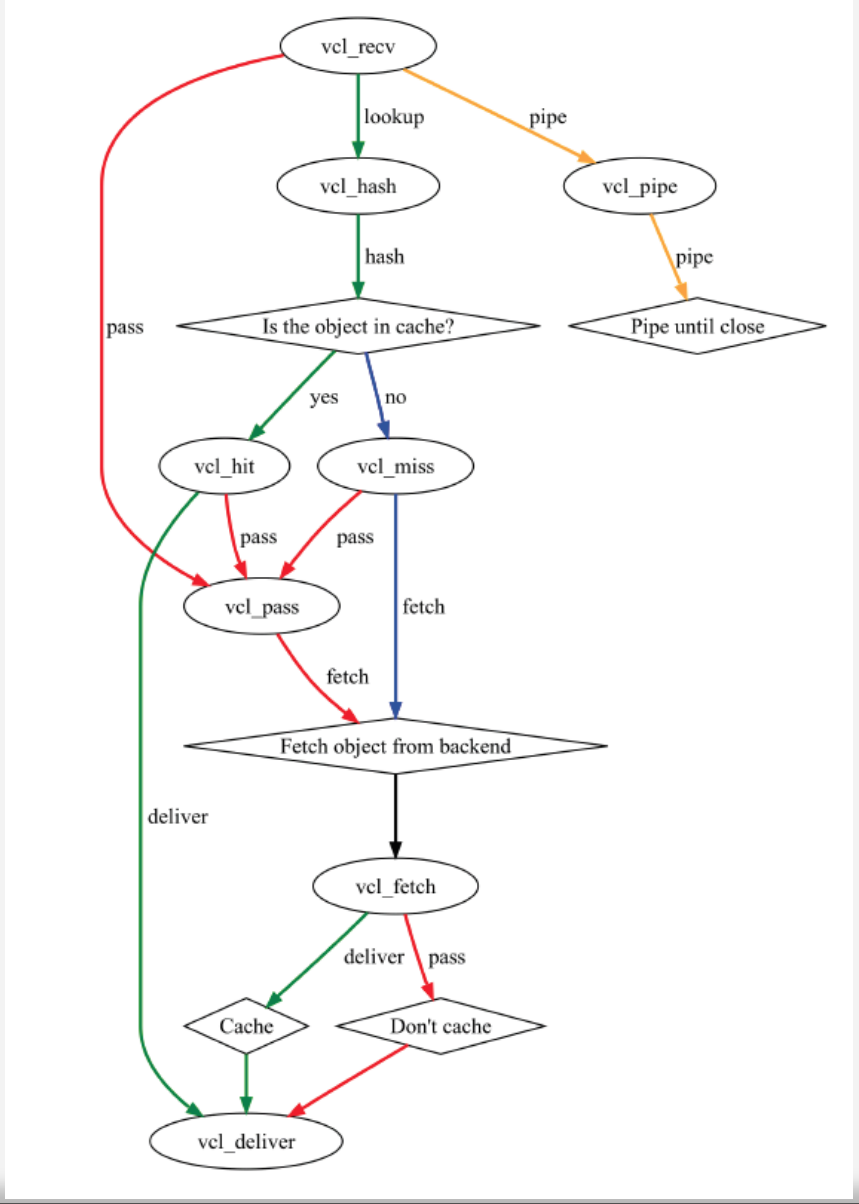
- *#* and */* foo */* for comments
- sub *$name* functions

Functions

- regsub(str, regex, sub) / regsuball(str, regex, sub)
- ban_hash(regex) / ban_url(regex) / ban (expression)
- restart
- return()

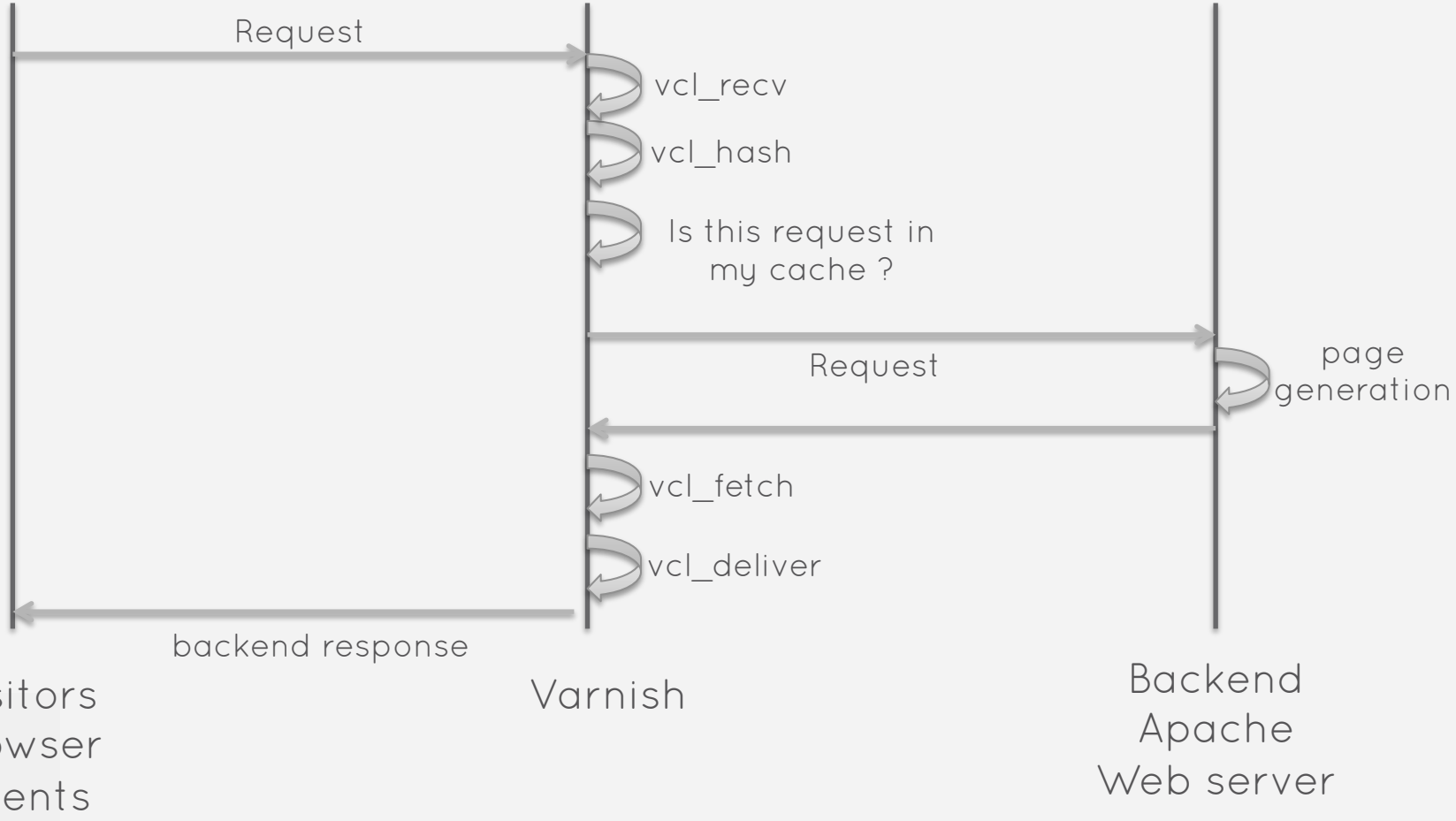
Based on a Request Flow

Simplified Varnish request flow



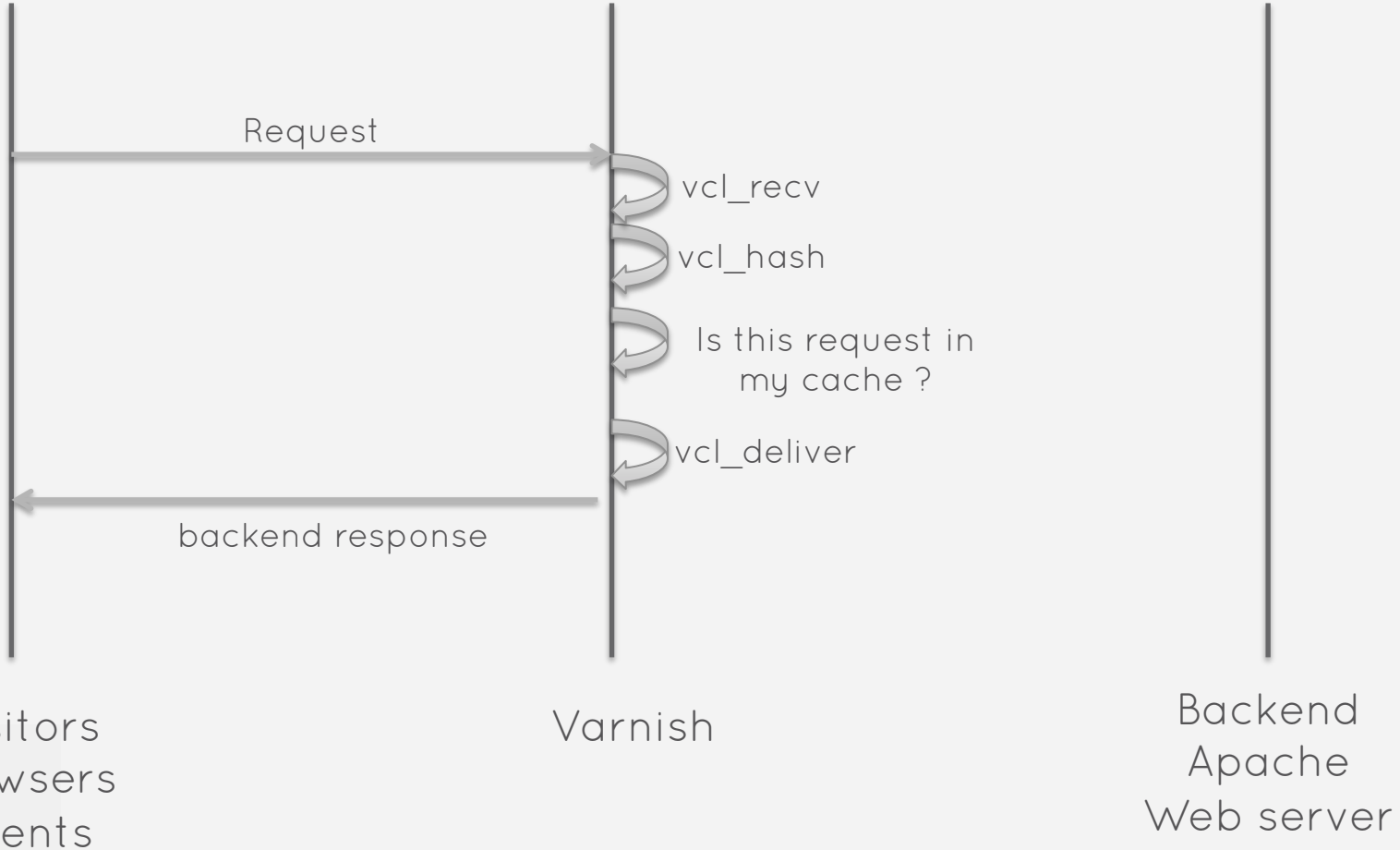
Varnish flow in diagram

Macro Simplified Varnish Flow : MISS



Varnish flow in diagram

Macro Simplified Varnish Flow : HIT



Varnish Flow

Recv

- Called at the beginning of a request, after the complete request has been received

Fetch

- Called when the requested object has been retrieved from the backend

Deliver

- Called before a cached object is delivered to the client

Hash

- Called to define the unicity of a cached object

Error

- Called when an error occurs

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- ***Varnish Advanced #1***
- ESI Basics
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Varnish advanced #1

VCL: Varnish Configuration Language

Debug information

- Usually added in the headers
- But, in the response headers !

HIT or MISS ?

- Most important information
- In the *vcl_deliver*

```
# Called before a cached object is delivered to the client
sub vcl_deliver {
    if (obj.hits > 0) {
        set resp.http.X-Cache = "HIT";
        set resp.http.X-Cache-Hits = obj.hits;
    } else {
        set resp.http.X-Cache = "MISS";
    }
    set resp.http.WhoisCache = "eZ Summer Camp Conf – Varnish 1"
    return(deliver);
}
```

- You should also add
  - the "hits" counter
  - the name of the Varnish server (if you are in a multi server architecture)

MISS ? Why ?

- Different ways to have a MISS
  - It's the first call after an expiration
  - Because it's a rule in the vcl_recv
    - return(pass)
  - Because it's a rule in the vcl_fetch
    - return(hit_for_pass)
- How to find out why you have a MISS
  - In the recv
    - Only access to the request (req)
  - In the fetch
    - You have access to the response (beresp)
- Solution ?
  - Add Header into the request in the recv function
  - Copy these headers into the response in the fetch function

MISS ? Why ?

- In the *vcl_recv*

```
# called at the beginning of a request, after the complete request has been received
sub vcl_recv {

… some rules …

if (req.request == "POST") {
    set req.http.X-Debug = "Not Cached according to configuration (POST)"; }
    return(pass);
}
```

- In the vcl_fetch

```
# Called when the requested object has been retrieved from the backend
sub vcl_fetch {

… some rules …

# show our debug
set beresp.http.X-Debug-recv = req.http.X-Debug;

… some rules …

# You are respecting the Cache-Control=private header from the backend
if ( beresp.http.Cache-Control ~ "private") {
    set beresp.http.X-Debug-fetch = "Not Cached according to configuration (Cache-Control)";
    set beresp.ttl = 0s;
    return(hit_for_pass);
}
```

Add ACL

- Everybody can see these headers
- Not very secure

```
# ACL for debug IP
acl debuggers {
    "localhost";
    "192.168.0.0"/16;
}
```

- Usage

```
# pass mode can't handle POST (yet) so use pipe
if (req.request == "POST") {
     if (client.ip ~ debuggers) {
         set req.http.X-Debug = "Not Cached according to configuration (POST)";
         return(pass);
     }
}
```

Add C in your VCL

- Useful to do very specific stuff

```
C{
     You C code
}C
```

- You have to see into the source code of Varnish to know the valid method

```
if( beresp.http.X-ttl ~ "s$"){ # seconds
    C{
         char *ttl;
         ttl = VRT_GetHdr(sp, HDR_BERESP, "\06X-ttl:");
         VRT_l_beresp_ttl(sp, atoi(ttl));
    }C
}
```

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ***ESI Basics***
- ESI with eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Edge Side Include
Definition

Edge Side Include

Definition

- markup language
  - not interpreted by the web server

```
<esi:include src="http://example.com/1.html"/>
```

- Inclusion of pages fragments
- Assembled on the edges
  - On reverse proxy (Varnish, Akamai, or other CDNs)

- Enable ESI in the VCL fetch method

```
set beresp.do_esi = true;
```

Edge Side Include

Be careful

- Do not parse ESI in images and binary content
- Self-closing XML element
- ESI will be ignored in pipe (see Varnish Flow)
- ESI requests are considered as isolated requests
  - ***Don't forget that !***

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ***ESI with eZ Publish***
- Recursive ESI
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

ESI with eZ Publish

A cache-block replacement

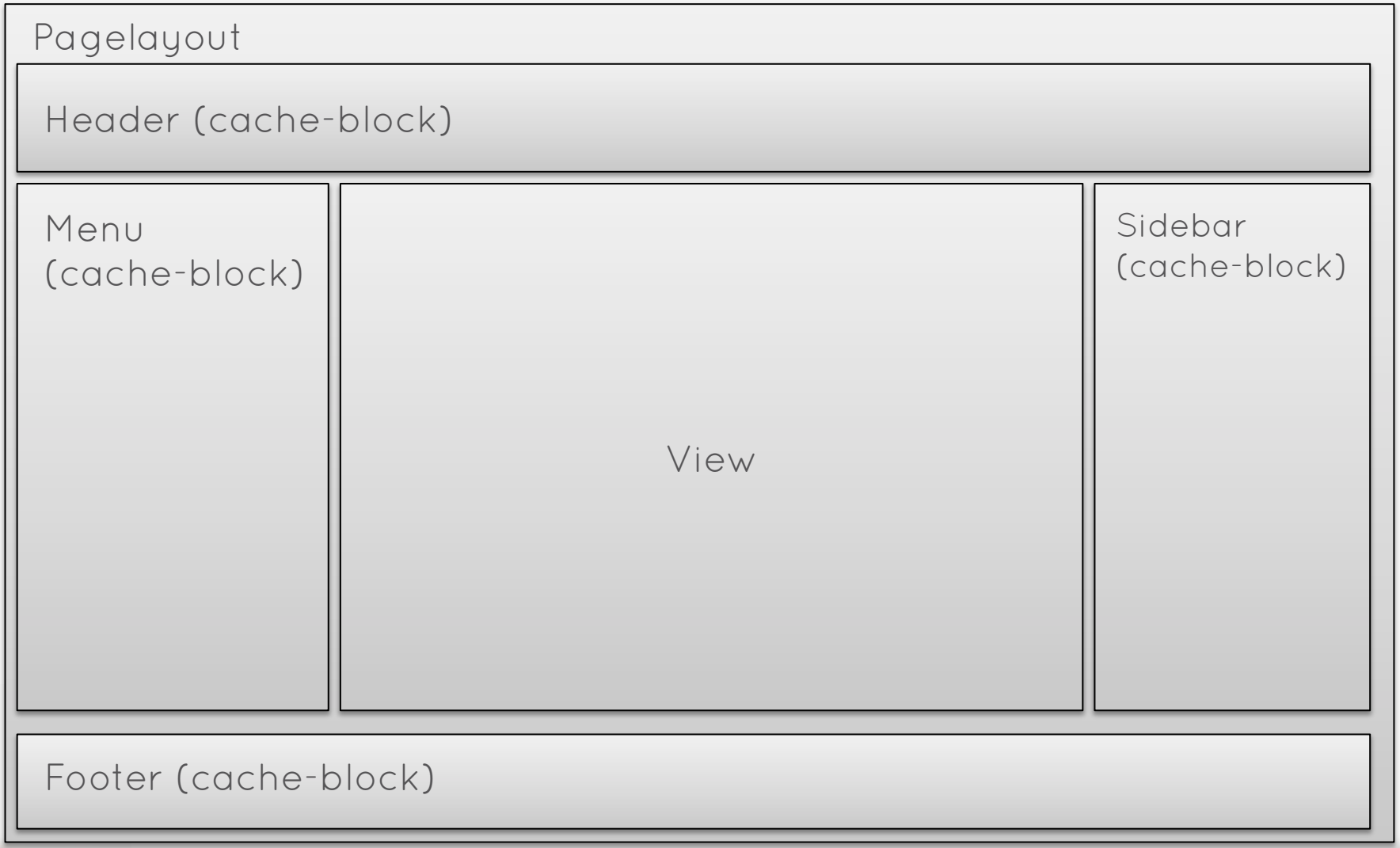
Objectives

- Managing cache-block
- Varnish became an applicative part of the platform
- ESI System
  - Deleting cache-block and using benefit of ESI views
- Update these caches (old cache -block) with the ezpublish view caching system.

- **Let's introduce the ESI content view...**

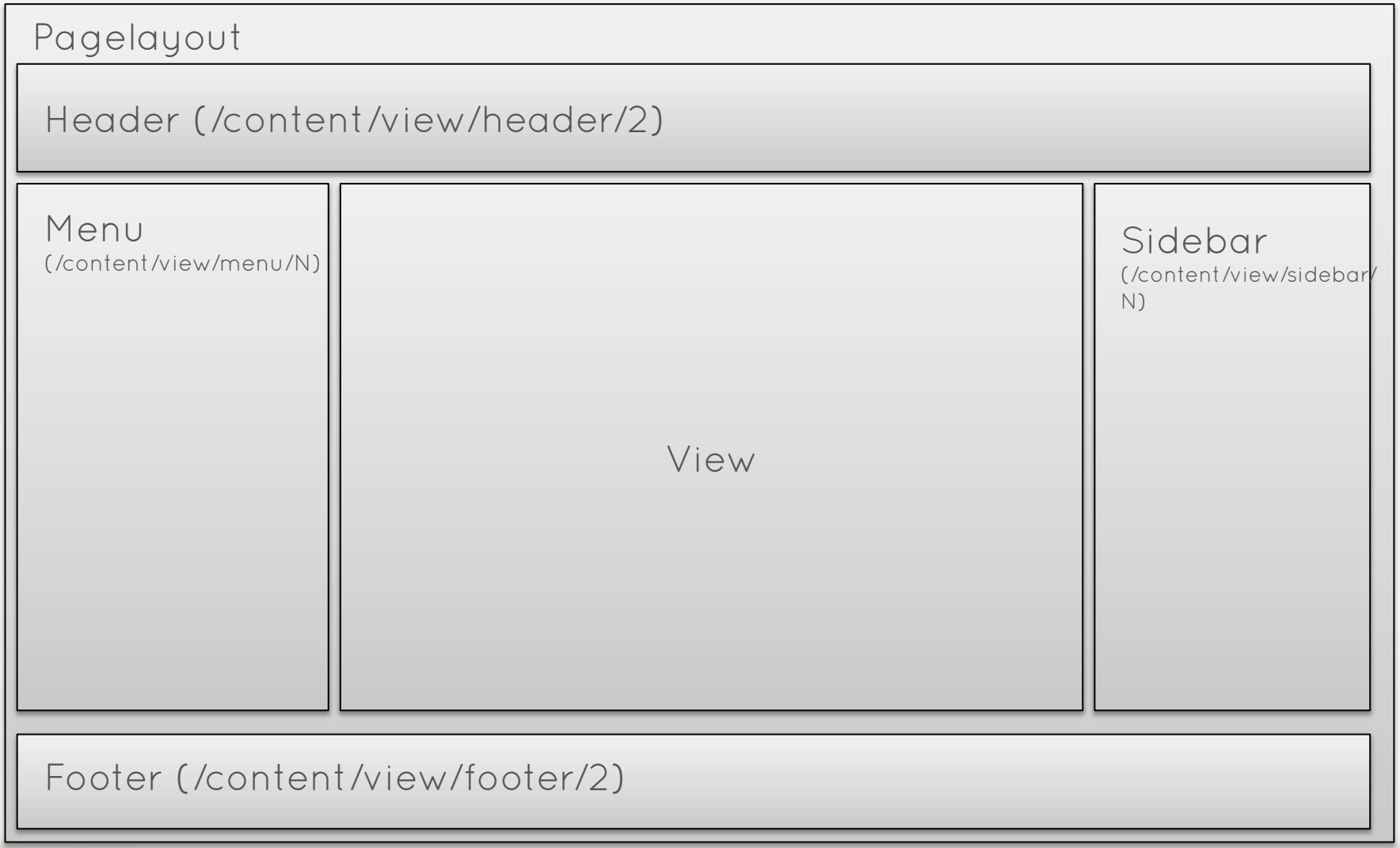
ESI with eZ Publish

Standard pagelayout : With cache-block



ESI with eZ Publish

ESI pagelayout: Without cache-block



Cache-block reminder performance and usage

- ignore_content_expiry
  - WITHOUT : each publication expires all your cache-block
  - WITH : you have to wait the TTL (expiry)
- subtree_expire
  - Not very useful in most case
  - Needs cronjob
- keys
  - two templates with same keys will generate at least 2 cache-block
- ***In general, cache-block can be a performance killer !***

ESI content view

- Forget the key concept of the cache-block
- Think about
  - content view
  - view cache system
  - smart view cache system
- For now, each ESI content view will have short TTL
  - But keep in mind, that it won't always be the case

ESI content view VS cache-block

- When the TTL is expired, there is no generation
  - You get the view cache
- You can easily clear a view
  - For now you have to wait the Varnish TTL
- You can use these views everywhere
  - In another view, or in the pagelayout !
- And if you want a TTL regeneration, don't add the view in your CacheViewModes

How to do better ?

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI with eZ Publish
- ***Recursive ESI***
- Varnish purge on eZ Publish publication
- Varnish Advanced #2

Recursive ESI with eZ Publish

Simplify your code

Recursive ESI with eZ Publish

Concept

- Varnish supports recursive ESI inclusion
- Limited by the *max_esi_includes* parameter
- Very useful when you have to browse a tree
- It's the one way to use ESI content view

Can we do better ? Again?

Yes !

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI for eZ Publish
- Recursive ESI
- ***Varnish purge on eZ Publish publication***
- Varnish Advanced #2

Varnish purge on eZ Publish publication

Unleash your TTL

Varnish purge on publication

Concept

- When eZ Publish expires a content view, forward this expiration to Varnish
- How ? 2 ways to send a purge request
  - Use the ezpEvent « content/cache »
  - Use a StaticCacheHandler
- Configure the vcl to handle this request and expire the cache object.
- Use the Smart View Cache to connect the different ESI view purge
- After that, you can increase your TTL !
  - To an unllimited value

Varnish purge on publication

First, how to handle a purge request

- In the vcl_recv

```
if (req.request == "BAN") {
    # ACL check and quick return (error)
    if (!client.ip ~ purgers) {
        error 405 "Method not allowed";
    }

    if (req.url ~ "^/url/") {
        ban_url(regsub(req.url, "^/url/", ""));
        error 200 "Purge of objects with url ("+regsub(req.url, "^/url/", "")+") done.";
    }
}
```

Summary

- Novactive
- Reverse Proxy
- Varnish eZ Publish Architecture
- Varnish Basics
- Varnish Advanced #1
- ESI Basics
- ESI for eZ Publish
- Recursive ESI
- Varnish purge on eZ Publish publication
- ***Varnish Advanced #2***

Varnish advanced #2

VCL advanced

Pool of backend

- random

```
director www_director random {
    {
          .backend = web1;
          .weight = 1;
    }
    {
           .backend = web2;
           .weight = 1;
    }
}
```

- Don't forget to add the second backend

```
backend web2 {
            .host = "localhost";
            .port = "80";
}
```

- Change the backend

```
set req.backend = www_director;
```

Check the heath of your backend

- Poke your backend every N seconds
- Affects backend selection

```
backend web1 {
    .host = "localhost";
    .port = "80";
    .probe = {
        .url = "/";
        .interval = 3s;
        .window = 5;
        .threshold = 2;
    }
}
```

- Varnish doesn't send Host with health checks.

```
.probe = {
    .request =
         "GET / HTTP/1.1"
         "Host: www.foo.bar"
         "Connection: close";
```

- Don't set the eZ Publish home page for the probe.

Check the heath of your backend

- req.grace (in the vcl_recv)
  - accept serving 30 minutes old object

```
# set the grace
set req.grace = 30m;
```

- beresp.grace (in the vcl_fetch)
  - keep all objects for 30 minutes beyond their TTL

```
# set the grace
Set beresp.grace = 30m;
```

- It's like the "Stale cache"

RewriteRule

- Varnish can do it

```
sub vcl_recv {

… some rules …

    if ( req.url ~ "^/layout/(.[^/]*)/([0-9]*)(.*)$" ) {
       set req.url = regsub(
                                    req.url,
                                    "^/layout/(.[^/]*)/([0-9]*)(.*)$",
                                    "/layout/set/\1/content/view/full/\2\3”
                        );
    }

… some rules …

}
```

Do a 301 HTTP redirection

```
sub vcl_recv {

… some rules …

if (req.http.host ~ "agreatbutlongdns.ezsc" ) {
    error 301 "ezpublish4.ezsc"; #see vcl_error
}
… some rules …

}

sub vcl_error {
    … some rules …
   if (obj.status == 301) {
       set obj.http.Location = "http://" + obj.response + ":" + server.port + req.url;
    } else {
        … some rules …
    }
}
```

Varnish advanced #2

Usage

Varnish advanced #2

varnishlog

```
3 TxHeader     c X-Varnish: 2075994459 2075993851
3 TxHeader     c Age: 20
3 TxHeader     c Via: 1.1 varnish
3 TxHeader     c Connection: close
3 TxHeader     c X-Cache: HIT
3 TxHeader     c X-Cache-Hits: 25
3 TxHeader     c WhoisCache: eZ Summer Camp Conf
3 Length       c 3371
3 ReqEnd       c 2075994459 1346209506.080837250 1346209506.080962420 0.000036001 0.000043392 0.000081778
3 SessionClose c Connection: close
3 StatSess     c 127.0.0.1 60969 0 1 1 0 0 0 589 3371
3 SessionOpen  c 127.0.0.1 60970 :8080
3 ReqStart     c 127.0.0.1 60970 2075994460
3 RxRequest    c GET
3 RxURL        c /Getting-Started/Resources/eZ-Publish-Tutorials
3 RxProtocol   c HTTP/1.1
3 RxHeader     c Host: ezpublish4.ezsc:8080
3 RxHeader     c Cookie: eZSESSID=re03avredomno9vlliqqj4qo71
3 RxHeader     c Accept: */*
3 RxHeader     c Accept-Encoding: gzip
3 RxHeader     c User-Agent: JoeDog/1.00 [en] (X11; I; Siege 2.70)
3 RxHeader     c Connection: close
3 VCL_call     c recv lookup
3 VCL_call     c hash
3 Hash         c /Getting-Started/Resources/eZ-Publish-Tutorials
3 Hash         c ezpublish4.ezsc:8080
3 VCL_return   c hash
3 Hit          c 2075993840
3 VCL_call     c hit deliver
3 VCL_call     c deliver
3 VCL_acl      c MATCH debuggers localhost
3 VCL_return   c deliver
3 TxProtocol   c HTTP/1.1
3 TxStatus     c 200
3 TxResponse   c OK
3 TxHeader     c Server: Apache/2.2.22 (Ubuntu)
3 TxHeader     c X-Powered-By: eZ Publish
3 TxHeader     c Expires: Mon, 26 Jul 1997 05:00:00 GMT
3 TxHeader     c Cache-Control: no-cache, must-revalidate
3 TxHeader     c Pragma: no-cache
3 TxHeader     c Last-Modified: Wed, 29 Aug 2012 03:04:45 GMT
3 TxHeader     c Served-by: ezpublish4.ezsc
3 TxHeader     c Content-language: en-GB
3 TxHeader     c Vary: Accept-Encoding
3 TxHeader     c Content-Encoding: gzip
3 TxHeader     c Content-Type: text/html; charset=utf-8
3 TxHeader     c Content-Length: 4503
3 TxHeader     c Accept-Ranges: bytes
```

varnishlog

- quite extensive
- filter information you need
- Useful to debug what happens
  - And valid your configuration

Best options

- **-o** : group by request
- **-b** : only show traffic related to a ***backend***
- **-c** : only show traffic related to a ***client***
- **-i <tag>** : only show one ***tag***

varnishtop

- Useful to know some real-time information
  - top URLs sent to the client
  - top URLs sent to the backend

varnishncsa

- It allows you to log client access as Apache do
- NCSA log file example

```
127.0.0.1 - - [29/Aug/2012:03:50:02 +0200] "GET http://ezpublish4.ezsc:8080/Getting-Started/Selected-
Features HTTP/1.1" 200 4314 "-" "JoeDog/1.00 [en] (X11; I; Siege 2.70)"
127.0.0.1 - - [29/Aug/2012:03:50:02 +0200] "GET http://ezpublish4.ezsc:8080/content/view/tagcloud/2
HTTP/1.1" 200 3371 "-" "JoeDog/1.00 [en] (X11; I; Siege 2.70)"
127.0.0.1 - - [29/Aug/2012:03:50:02 +0200] "GET http://ezpublish4.ezsc:8080/content/view/sitemap/2
HTTP/1.1" 200 3187 "-" "JoeDog/1.00 [en] (X11; I; Siege 2.70)"
127.0.0.1 - - [29/Aug/2012:03:50:02 +0200] "GET http://ezpublish4.ezsc:8080/content/advancedsearch
HTTP/1.1" 200 4505 "-" "JoeDog/1.00 [en] (X11; I; Siege 2.70)"}
```

varnishstat

- Perfect tool to get a good representation of your Varnish health
- Some useful statistic
  - Uptime
  - HIT rate
  - Client connections accepted
  - Client connections received
  - Cache HIT
  - Cache MISS
  - Failed backend connections

Varnish advanced #2

varnishstat

```
0+00:21:36
Hitrate ratio:        10      100      147
Hitrate avg:     0.9559   0.9703   0.9711

        4928        28.97         3.80 client_conn - Client connections accepted
        4953        26.97         3.82 client_req - Client requests received
        4585        24.98         3.54 cache_hit - Cache hits
         169         0.00         0.13 cache_miss - Cache misses
          37         0.00         0.03 backend_conn - Backend conn. success
         333         4.00         0.26 backend_reuse - Backend conn. reuses
          32         0.00         0.02 backend_toolate - Backend conn. was closed
         368         2.00         0.28 backend_recycle - Backend conn. recycles
         368         2.00         0.28 fetch_length - Fetch with Length
          16            .            .   n_sess_mem - N struct sess_mem
           2            .            .   n_sess - N struct sess
         169            .            .   n_object - N struct object
         175            .            .   n_objectcore - N struct objectcore
          54            .            .   n_objecthead - N struct objecthead
          11            .            .   n_waitinglist - N struct waitinglist
           5            .            .   n_vbc - N struct vbc
           6            .            .   n_wrk - N worker threads
           6         0.00         0.00 n_wrk_create - N worker threads created
          53         0.00         0.04 n_wrk_queued - N queued work requests
           2            .            .   n_backend - N backends
         953            .            .   n_lru_moved - N LRU moved objects
        4953        26.97         3.82 n_objwrite - Objects sent with write
        4926        26.97         3.80 s_sess - Total Sessions
        4953        26.97         3.82 s_req - Total Requests
```

Varnish advanced #2

- Find the URL the most sent to the backend

```
Varnistop –i TxURL
```

- Find the URL the most asked by the client

```
Varnistop –i TxURL
```

- Have you try the « -j » option of *varnishstats*
  - new in 3.03 since 08/20/12

Varnish advanced #2

Daemon configuration

Varnish advanced #2

BAN Console

- Access
  - varnishadm
  - telnet localhost 6082
- Tunable paramaters
  - param.show -l
- Be careful about the thread parameters

```
root@ezsc ~ # varnishadm param.show | grep thread
cc_command                    "exec gcc -std=gnu99 -g -O2 -pthread -fpic -shared -Wl,-x -o %o %s"
thread_pool_add_delay         2 [milliseconds]
thread_pool_add_threshold     2 [requests]
thread_pool_fail_delay        200 [milliseconds]
thread_pool_max               2000 [threads]
thread_pool_min               1 [threads]
thread_pool_purge_delay       1000 [milliseconds]
thread_pool_stack             65536 [bytes]
thread_pool_timeout           120 [seconds]
thread_pool_workspace         16384 [bytes]
thread_pools                  2 [pools]
thread_stats_rate             10 [requests]
```

Threading parameters

- Number of threads
  - how many requests Varnish can serve concurrently
- Minimum threads running = *thread_pools* * *thread_pool_min*
  - 2 * 1 = 2
- Maximum threads running = *thread_pool_max*
  - 2000
- Don't forget to add your configuration in the default configuration
  - /etc/default/varnish

Real-time configuration changes

- you can change a param with *varnishadm*
  - *but don't forget to add the new parameter value in the daemon file configuration*
- About the VCL, be careful because a restart empties the cache
  - Varnish don't keep cache
  - very bad idea in a production environment !
- So let's use *varnishadm* in order to reload the VCL

To sum up...

To sum up

On the network administration side, you know

- How to set architecture of your plateform
- How to configure your Varnish
- How to optimize the daemon
- How to optimize your VCL
- How to troubleshoot your instance

To sum up

On the developer side, you know

- How to make a good conception with ESI content view
- How to remove cache-block and unleash your instance
- How to purge cache on publication

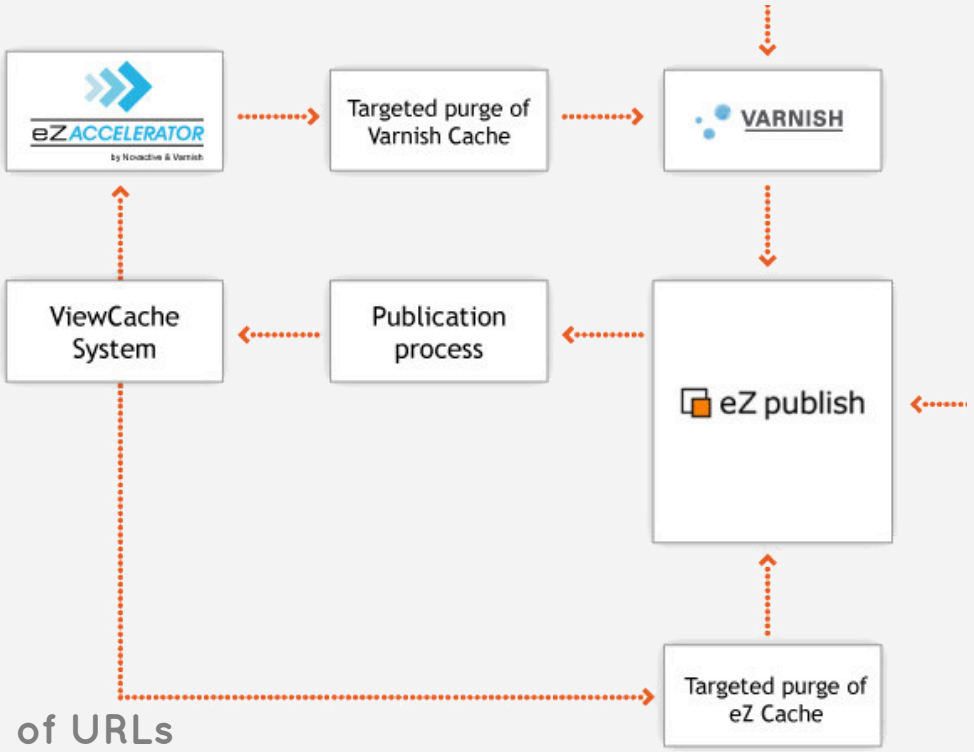
Conclusion

Achieved objectives

- You can now have a production website with
  - 98% of requests cached by Varnish
  - 2 % of traffic handle by your webserver farm
- Set a architecture with
  - A database dedicated to the backoffice and to the generation on new content
  - A very low usage of your NFS
- Beyond speed, more security
  - Your website is all the time available due to the shield protection of Varnish
  - even if we have server problems

eZ Accelerator

To go further, discover eZ Accelerator available on the eZ Market



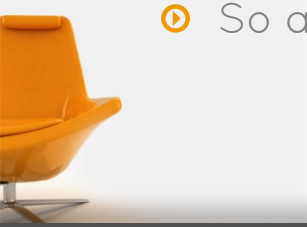
Key Features

- Full support of URLs
  - URL Translator, Siteaccess, PathPrefix, StaleCache etc..
- Asynchronous purge
- Multi varnish architecture
- Cache purge on content publication
- Dedicated interface to manage your Varnish servers

eZ ACCELERATOR
by Novactive & Varnish

eZ Publish 5 with Varnish

- Varnish on Symfony 2
  - Totally integrated with the HTTP Cache system
  - ESI are directly handled by the template and cache system
- Two caching models
  - Expiration model
    - Based on the TTL
  - Validation model
    - Based on eTag
- Always the same rule : Never generate the same response twice !
- So all the concept seen on this workshop are relevant with eZ Publish 5



Questions ?

Thank you!

novactive

Adresse : 42-44, rue de Paradis
75010 PARIS

Téléphone : 01.48.24.33.60
Fax : 01.48.24.33.54

Email : info@ novactive.com
Site : www.novactive.com

SARL au capital de 132.576 euros
RCS Paris B 408 999 233