Cloudy-cloud-enabled

Dynamic backends in Varnish 4.1

About me

- Developer
  - I contribute features, documentation and mostly bugs
  - I hate the web
  - I love Varnish
- French
  - ~~Your problem, not mine~~

More about me

- `Dec 2011:` I discover Varnish 3, I stop doing Java
- `Jun 2012:` I start my first VMOD
- `Dec 2012:` I give my first Varnish training
- `Apr 2014:` My first patch lands in Varnish
- `Mar 2015:` I join Varnish Software
- `Oct 2015:` First blog post at Varnish Software
- `Dec 2015:` My first VUG \o/

Two* things I learned

- There is almost always a caching requirement behind each functionality
- Simplicity is almost always preferred given more than one obvious solution
- Knowing the history of the project helps better understand its current state

The VBE and VDI subsystems

The VBE and VDI subsystems

- Back to basics

```
    ______           _________           _________
   |      |         |         |         |         |
   |______| -----> | Varnish | -----> | Backend |
  /______/          |_________|         |_________|
```

The VBE and VDI subsystems

- Back to basics

```
                                     _________
                                    |         |
                               +--> | Backend |
                               |    |_________|
  ______          _________    |     _________
 |      |        |         |   |    |         |
 |______| -----> | Varnish | --+--> | Backend |
/______/         |_________|   |    |_________|
                               |     _________
                               |    |         |
                               +--> | Backend |
                                    |_________|
```

The VBE and VDI subsystems

- In Varnish 3, tight coupling with VCL

```
backend www {
  .host = "www.example.com";
  .port = "http";
}
```

```
director dir random {
  .retries = 5;
  {
    // We can refer to named
    // backends
    .backend = fs1;
    .weight  = 7;
  }
  {
    // Or define them inline
    .backend = {
      .host  = "fs2";
    }
    .weight  = 3;
  }
}
```

The VBE and VDI subsystems

- In Varnish 4, directors removed from VCL

```
backend www {
  .host = "www.example.com";
  .port = "http";
}
```

```
import directors;

sub vcl_init {
  new dir =
    directors.round_robin();
  dir.add_backend(b1);
  dir.add_backend(b2);
}

sub vcl_recv {
  set req.backend_hint =
    dir.backend();
}
```

The VBE and VDI subsystems

- New capabilities
- But no more DNS director

```
                                        __________
                                       |          |
                                  +--> | Director | --> ...
                                  |    |__________|
 _________          __________    |     __________
|         |        |          |   |    |          |
| Varnish | -----> | Director | --+--> | Director | --> ...
|_________|        |__________|   |    |__________|
                                  |     __________
                                  |    |          |
                                  +--> | Director | --> ...
                                       |__________|
```

The VBE and VDI subsystems

- Cloudy-cloud-enabled
  - VCL live-reload is immediate
  - Backends are static, VCL is dynamic

- How to cloud?
  - Listen to infrastructure events
  - Generate VCL
  - Reload VCL
  - Profit

Current state in 4.1

Current state in 4.1

- Changes in the director API
  - Native backends sort-of removed from VCL
    - VCL consumes the backend API
    - Same API available to VMODs
    - They can be dynamic as well as static
  - “Cluster” directors still available
    - Backend resolution chain handled by the VDI
  - “Custom” backends
    - No need to speak HTTP/1 over TCP on the backend side
    - Intermediary web servers can virtually disappear

Current state in 4.1

- Changes in the director API

```
/* Varnish 4.0 */

struct director {
  unsigned      magic;
  const char    *name;
  char          *vcl_name;
  vdi_getfd_f   *getfd;
  vdi_healthy   *healthy;
  void          *priv;
};
```

```
/* Varnish 4.1 */

struct director {
  unsigned          magic;
  const char        *name;
  char              *vcl_name;
  vdi_healthy_f     *healthy;
  vdi_resolve_f     *resolve;
  vdi_http1pipe_f   *http1pipe;
  vdi_gethdrs_f     *gethdrs;
  vdi_getbody_f     *getbody;
  vdi_getip_f       *getip;
  vdi_finish_f      *finish;
  vdi_panic_f       *panic;
  void              *priv;
  const void        *priv2;
};
```

Current state in 4.1

- Custom backends in a nutshell
  - If you write a VMOD, you can:

```
                                                +--> /var/www
                                                |    vmod-fsbackend
      ______          _________                 |
     |      |        |         |                |
     |______| -----> | Varnish | --+--> php-fpm
    /______/         |_________|                |
                                                |
                                                |
                                                +--> JServ (AJP)
                                                |
                                                |
                                                |
                                                +--> ...
```

Current state in 4.1

- Back to “VBE” backends
  - New VRT functions
    - `VRT_new_backend`
    - `VRT_delete_backend`
  - Create and delete them at any time*
    - No need to reload VCL
  - Don't delete backends you don't own
    - VCL backends are statically referenced for instance

DNS-based backends

DNS-based backends

- Problem to solve
  - Reload backends without reloading VCL
  - Infrastructure and cache policy have different life cycles

- Proposed solution
  - Rely on DNS to get a list of backends
  - ???
  - Profit

DNS-based backends

- Doesn't work with “VBE” backends

```
# VCL may not compile

backend google {
  .host = "google.com";
  .port = "http";
}
```

DNS-based backends

- Doesn't work with “VBE” backends
- A host name can resolve at most
  - One IPv4 address
  - One IPv6 address
- Varnish assumes both point to the same server

DNS-based backends

- Varnish 3
  - Create static backends
  - Enable/disable them according to look-ups

```
director directorname dns {
        .list = {
                .host_header = "www.example.com";
                .port = "80";
                .connect_timeout = 0.4s;
                "192.168.15.0"/24;
                "192.168.16.128"/25;
        }
        .ttl = 5m;
        .suffix = "internal.example.net";
}
```

DNS-based backends

- Varnish 4
  - Nothing
  - Might be possible though

DNS-based backends

- Varnish 4.1
  - https://github.com/dridi/libvmod-named
  - Turns Varnish into a forward proxy...

```
probe healthcheck { ... }

sub vcl_init {
  new dir = named.director("http");
  dir.probe_with(healthcheck);
  dir.set_ttl(5m);
}

sub vcl_recv {
  set req.backend_hint =
        dir.backend("www.example.com");
}
```

DNS-based backends

- Design differences
  - Static vs dynamic backends
  - Probe support in vmod-named
  - “White-list” support in the DNS director*
    - Not easily feasible in Varnish 4.1.0
    - Could be done by reusing ACLs
    - But my ACL patch missed the release window
  - Look-ups block workers in the DNS director
    - Look-ups are done in a separate thread in vmod-named
  - Production-ready vs PoC

Cloudy-cloud-enabled

Cloudy-cloud-enabled

- Varnish can now be aware of backends changes
- No need to reload the VCL for that
- Just sit back and relax
- Right?

Cloudy-cloud-enabled

- New capabilities, new caveats

- VCLs own backends instead of sharing
  - Because of dynamic backends*
  - Backends share connection pools

- A discarded VCL can stick for a long time*

Cloudy-cloud-enabled

- Let's play “spot the differences”

```
varnishstat -1 -f VBE.\* |
awk '/vcls|happy/ {print $1 "\t" $2}'
```

```
/* Varnish 4.0 */

VBE.default(,::1,80).vcls  10
VBE.default(,::1,80).happy 0
```

```
/* Varnish 4.1 */

VBE.vcl1.default.happy  0
VBE.vcl2.default.happy  0
VBE.vcl3.default.happy  0
VBE.vcl4.default.happy  0
VBE.vcl5.default.happy  0
VBE.vcl6.default.happy  0
VBE.vcl7.default.happy  0
VBE.vcl8.default.happy  0
VBE.vcl9.default.happy  0
VBE.vcl10.default.happy 0
```

VCL Temperature

VCL Temperature

- VCL can be cold or warm
  - A VCL must be warmed up before use
  - It can eventually cool down after use
  - New knobs to play with

- Overall a breaking change
  - Plan your upgrade first
  - I mean it!

VCL Temperature

- A cold VCL
  - Should not get in the way of the active VCL
  - Should have the lowest possible footprint
  - Will drop native backends probes and stats
- Remember that
  - Backends will grow “linearly”
  - The more the probes, the more the overhead
  - The stats segment is limited

VCL Temperature

- Design is however not complete
  - VMODs may get the information too late
  - But creating a “cold” backend is not allowed
  - VCLs may warm up before completely cooling down
  - Known problem, a patch is ready*

- Currently
  - Dynamic backends work
  - But only in sub vcl_init{}
  - A workaround is to only have warm VCLs

Cloudy-cloud-enabled

Cloudy-cloud-enabled

- Better suited for backends in the cloud
  - VCL reloads have more constraints
  - But backends can be refreshed any time
  - Decouple infrastructure and policy
  - Integrate with your favorite stack
  - Expect more from future releases!

- A lot more powers to VMOD writers

Thank you for your time!
Questions?

Check out 4.1 VMODs on github:
- mbgrydeland/libvmod-fsbackend
- dridi/libvmod-named