Reverse proxy automation - Varnish as a Service

Tomasz Ziółkowski 26.06.2018

Overview

allegro Tech

Reverse Proxy

- Forward vs Reverse
- Features
- Caching

Varnish

- Design
- Architecture
- State machine

Allegro use case

- Some metrics
- Monolith to SOA
- ESI

Overview

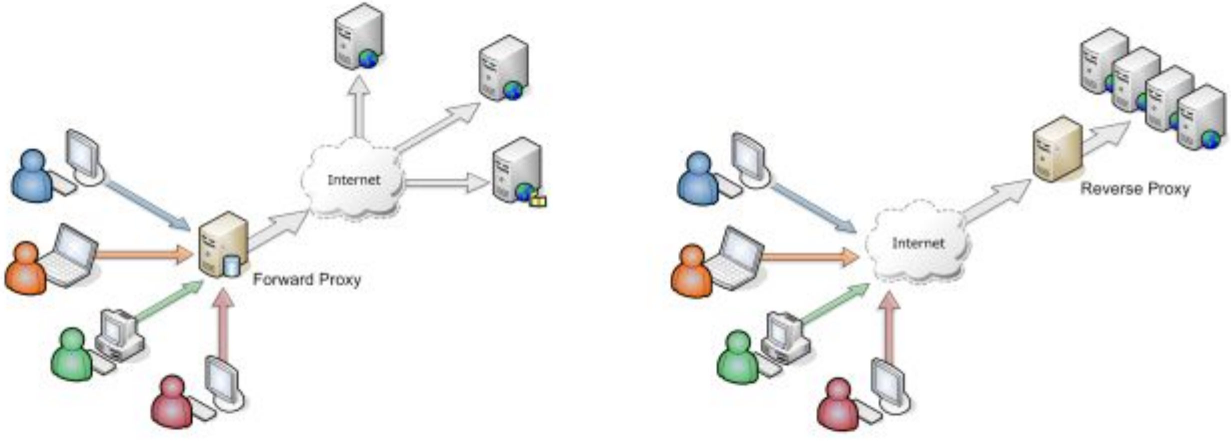
allegro Tech

VaaS

- Overview
- Automation
- Contribute :)

Reverse Proxy

Proxy - reverse vs. forward



Reverse proxy - features

Hide architecture

Hide sophisticated architecture of microservices

Access control on HTTP layer

L7 ACL based on http headers or client ip

SSL termination

Terminate SSL connection before origin servers

Reverse proxy - features

Loadbalancing

Balance traffic based on several predefined LB algorithms

Link usage optimization

Compress content for a client (if needed)

ESI

Composite html responses for a client from several origin servers responses

Reverse proxy - features

Caching

Content caching based on Cache-Control response headers

Availability

Active checks

Scalability

Reconfigure proxy after autoscaling

Reverse proxy - caching

Cache-Control response header

Caching levels

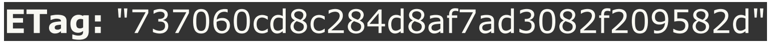
- public
- private
- no-cache
- no-store

```
Cache-Control: public,max-age=420;
```

Reverse proxy - caching

ETag response header

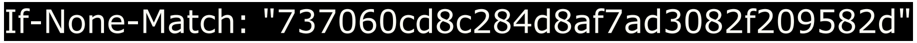
The ETag HTTP response header is an identifier for a specific version of a resource. It allows caches to be more efficient, and saves bandwidth, as a web server does not need to send a full response if the content has not changed.



```
ETag: "737060cd8c284d8af7ad3082f209582d"
```

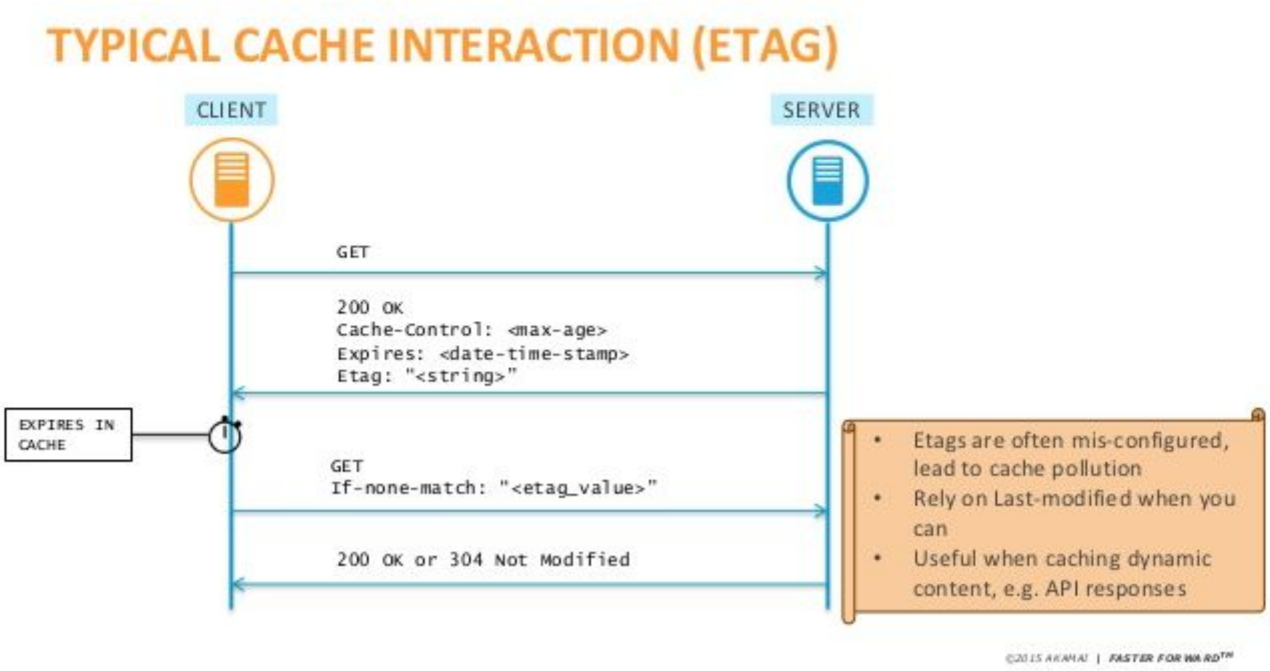
If-None-Match response header

Check if outdated resource are really modified.



```
If-None-Match: "737060cd8c284d8af7ad3082f209582d"
```

Reverse proxy - caching



TYPICAL CACHE INTERACTION (ETAG)

GET

200 OK
Cache-Control: <max-age>
Expires: <date-time-stamp>
Etag: "<string>"

GET
If-none-match: "<etag_value>"

200 OK or 304 Not Modified

- Etags are often mis-configured, lead to cache pollution
- Rely on Last-modified when you can
- Useful when caching dynamic content, e.g. API responses

©2015 AKAMAI | FASTER FORWARD™

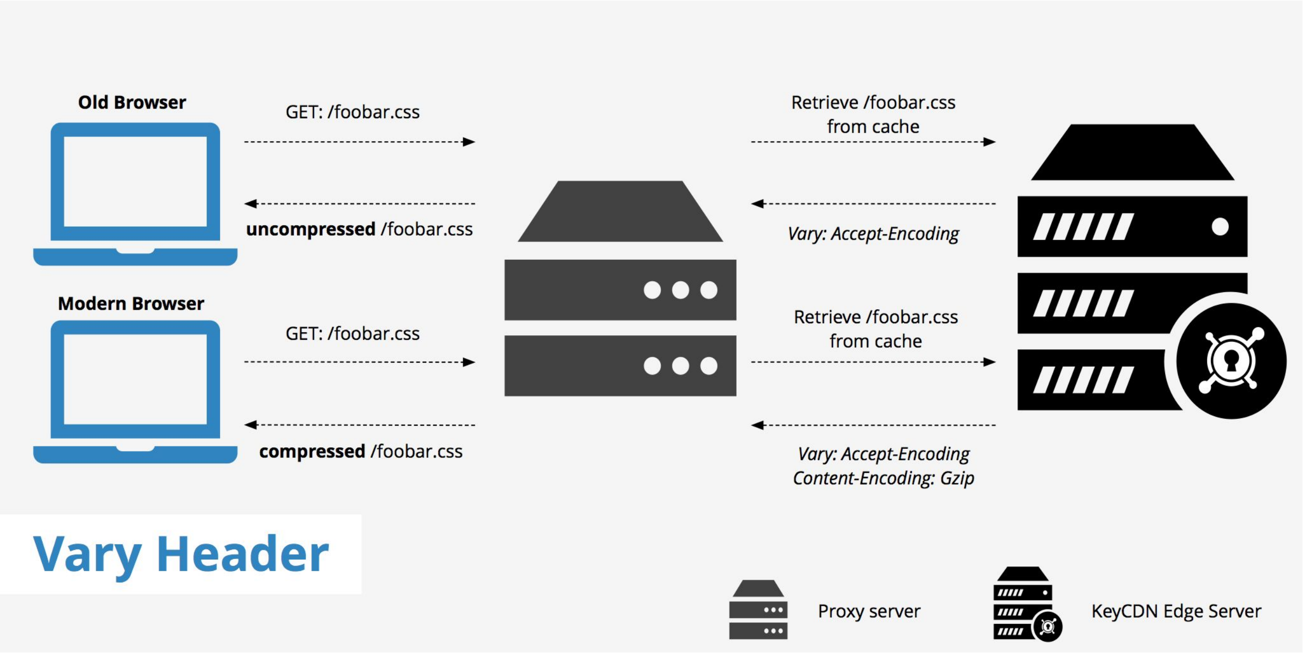
Reverse proxy - caching

Vary response header

The Vary HTTP response header determines how to match future request headers to decide whether a cached response can be used rather than requesting a fresh one from the origin server. It is used by the server to indicate which headers it used when selecting a representation of a resource in a content negotiation algorithm.

```
Vary: Accept-Encoding
```

Reverse proxy - caching



Varnish

Varnish - design

Designed as reverse-proxy

Handle only HTTP traffic

Configurable via DSL

No native API (CLI via telnet)

Varnish - terminology

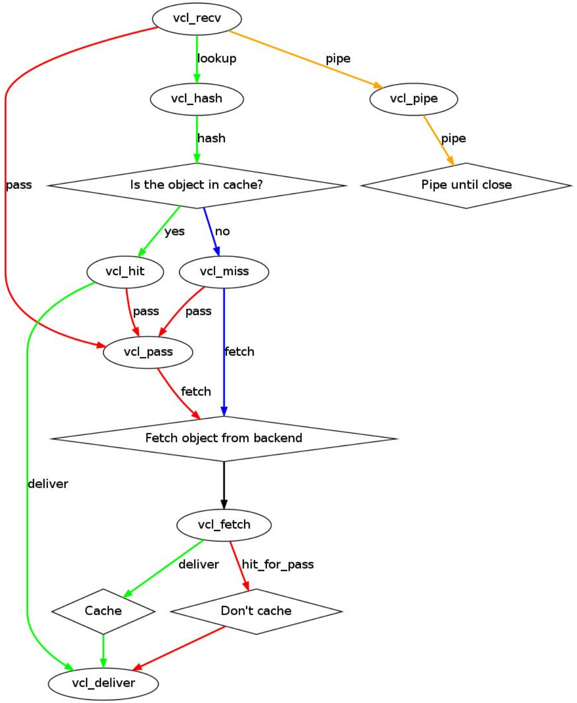
Backend - ip:port - origin server

Director - logical set of backends enriched with LB algorithms (RR, Random, Hash)

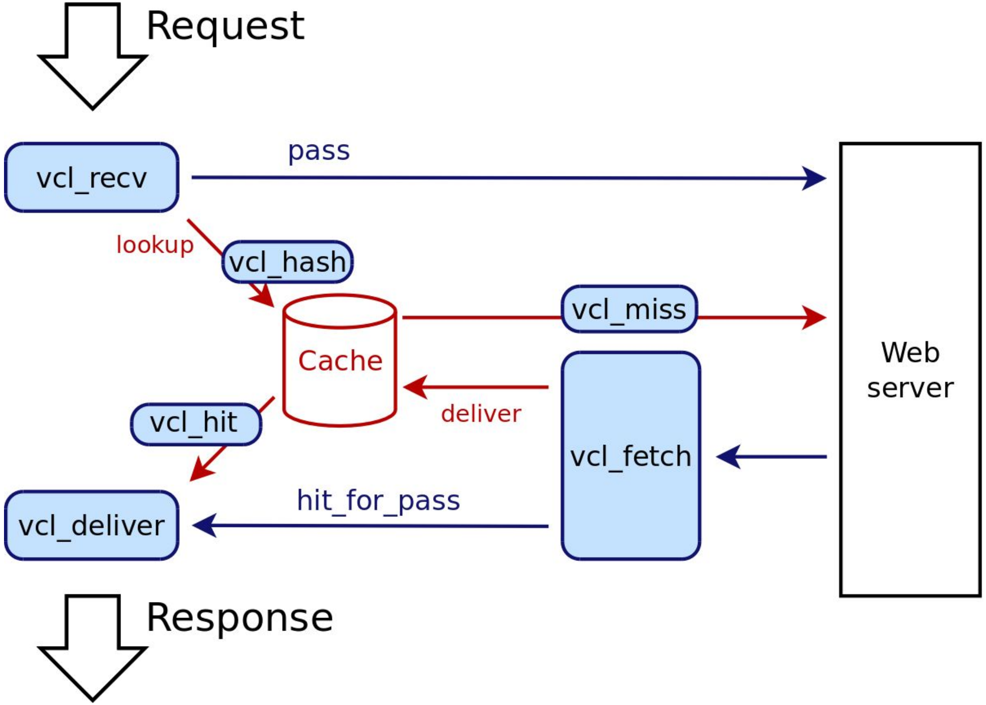
Probe - verify backend readiness to handle traffic

VCL - Varnish Configuration Language

Varnish - state machine



Varnish - simple flow



Varnish

Possibility to expand state machine

Possibility to modify http headers req/resp bereq/beresp

Possibility to use simple logical conditions

Varnish - VCL

Directors/backend definition

Predefined sub implementation:

- Recv
- Fetch
- Deliver

Custom sub implementation

Varnish - VCL example

```
backend default_backend {
  .host = "127.0.0.1";
  .port = "8888";
}

backend allegro {
  .host = "allegro.pl";
  .port = "80";
}
```

```
sub vcl_recv {
    if (req.http.host == "allegro.pl") {
        set req.backend_hint = allegro;
    } else {
        set req.backend_hint = default_backend;
    }
}

sub vcl_deliver {
    set resp.http.X-Hit = "HIT " + obj.hits;
}
```

Allegro use case

allegro.pl - some metrics

20k rps

5Gbps

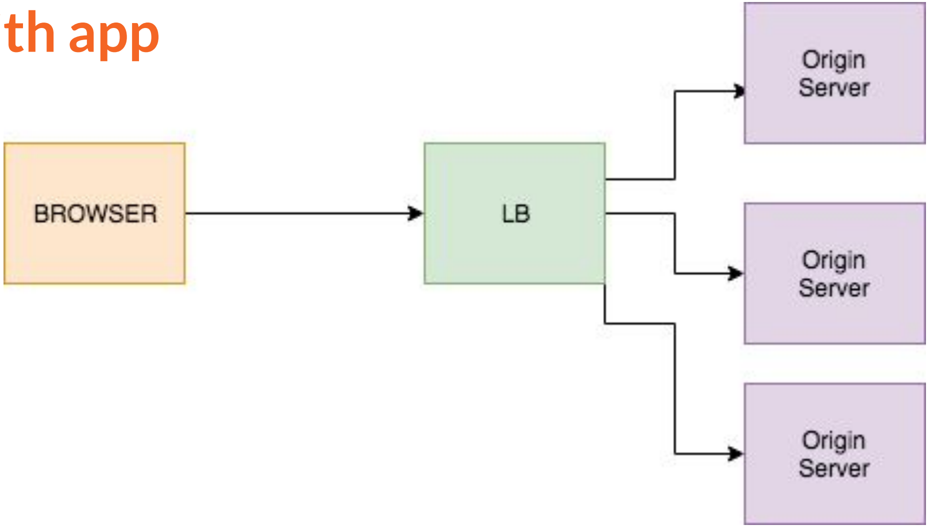
>100 front microservices

>800 origin servers

allegro Tech

Allegro - monolith to SOA

Old http architecture / monolith app



Allegro - monolith to SOA

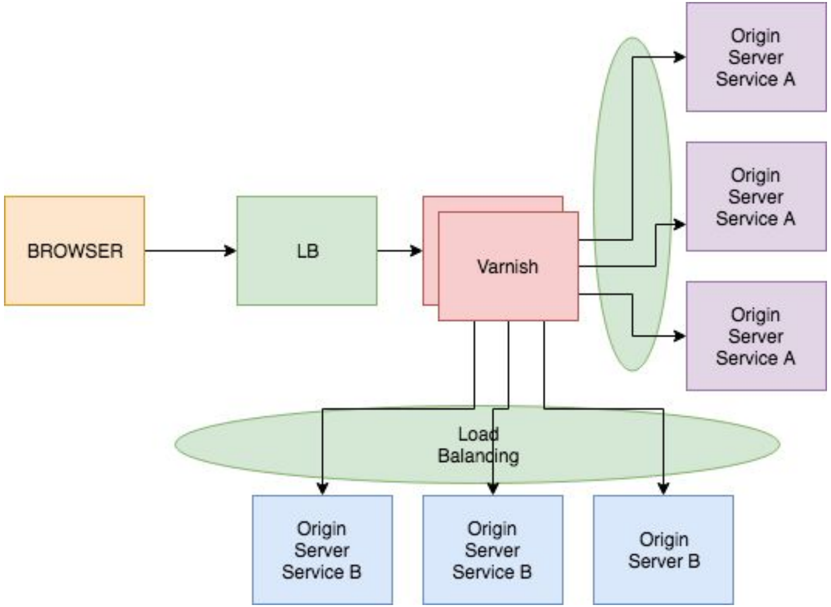
Challenges:

- Application routing to multiple microservices
- Composite single html response based on several microservices
- Unhealthy instances - active detection
- Load balancing based on cookies
- Retries for 5xx response codes
- Move cache near to client

allegro Tech

Allegro - monolith to SOA

Http architecture - varnish / SOA

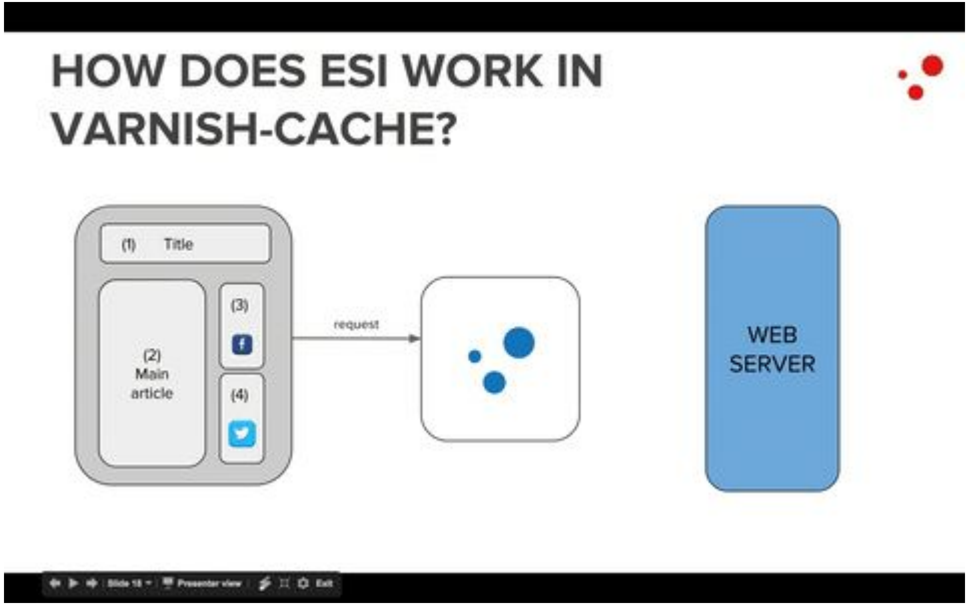


Allegro - ESI

Composite single html response:

- In main response use <esi src=”/path/to/fragment”>
- Varnish parse response and execute subrequest for each ESI tag
- ESI tags in main response are replaced by responses from subrequests

Allegro - ESI



HOW DOES ESI WORK IN VARNISH-CACHE?

Vaas

varnish as a service

VaaS - motivation

Monolith to SOA

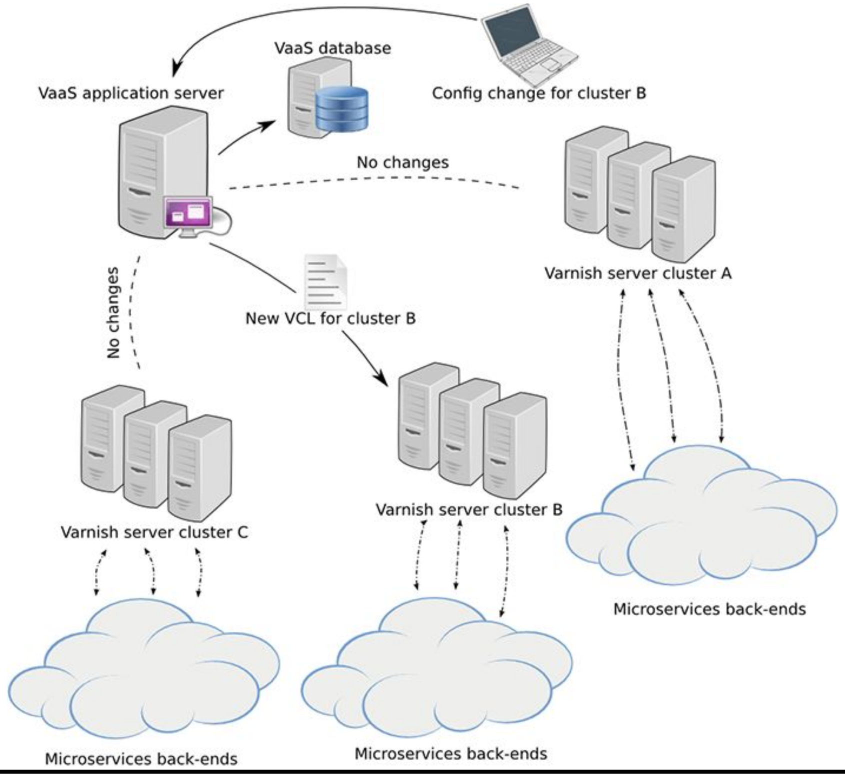
Lack of management tools

Reload configuration on each change

Automated reload after automated change

Automated support for local processing

VaaS - overview



VaaS - automation

Generating VCL based on templates & DB objects representing directors and backends

Homogeneous clusters (same configuration on each machine in the same dc)

Directors & backends exists independently of VCL

VaaS - automation

Automated generating VCL regard to local processing

Synchronous reconfiguration

Expose REST API - enable automation with other ecosystem components (ie. service-discovery)

VaaS - automation - Templates

Placeholders:

- VCL
- DIRECTORS
- ROUTER
- SET_BACKEND_director_name

VaaS - automation - Templates

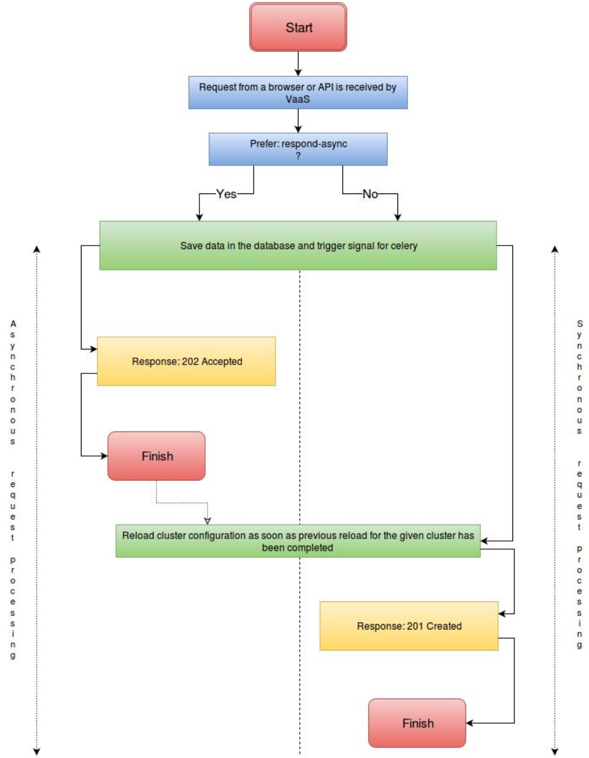
```
<VCL>
    <HEADERS/>
    <ACL/>
    <DIRECTORS>
        <DIRECTOR_{DIRECTOR}>
            <BACKEND_DEFINITION_LIST_{DIRECTOR}_{DC}/>
            <DIRECTOR_DEFINITION__{DIRECTOR}_{DC}>
                <BACKEND_LIST_{DIRECTOR}_{DC}/>
            </DIRECTOR_DEFINITION__{DIRECTOR}_{DC}>
        </DIRECTOR_{DIRECTOR}_{DC}>
        …
    </DIRECTORS>
    <RECV>
        <PROPER_PROTOCOL_REDIRECT/>
        <ROUTER>
            <SET_BACKEND_{DIRECTOR}/>
            ...
            <SET_BACKEND_{DIRECTOR}/>
        </ROUTER>
    </RECV>
    <OTHER_FUNCTIONS/>
</VCL>
```

VaaS - automation

VaaS is used by other parts of ecosystem via REST API

VaaS provide sync and async model of making changes

VaaS - automation


Start
Request from a browser or API is received by VaaS
Prefer: respond-async ?
Yes
No
Save data in the database and trigger signal for celery
Asynchronous request processing
Synchronous request processing
Response: 202 Accepted
Finish
Reload cluster configuration as soon as previous reload for the given cluster has been completed
Response: 201 Created
Finish

VaaS - automation

- Application routing - SOA
- Front microservices deployment
- Canary
- Redirect protocols
- Composite site from fragments
- HA, cross-dc fallbacks

… any many more is easier thanks to Varnish+VaaS

VaaS numbers - in allegro

- 20 clusters
- >70 varnishes
- >200 directors
- >1000 backends
- 50-100 deployments per day
- Thousands of VCL changes per day

VaaS - allegro dev perspective

- Integrated with sophisticated ecosystem
- Transparent for most operations
- Similar to declarative infrastructure as LoadBalancer in K8s

VaaS - contribute :)

Prerequisite

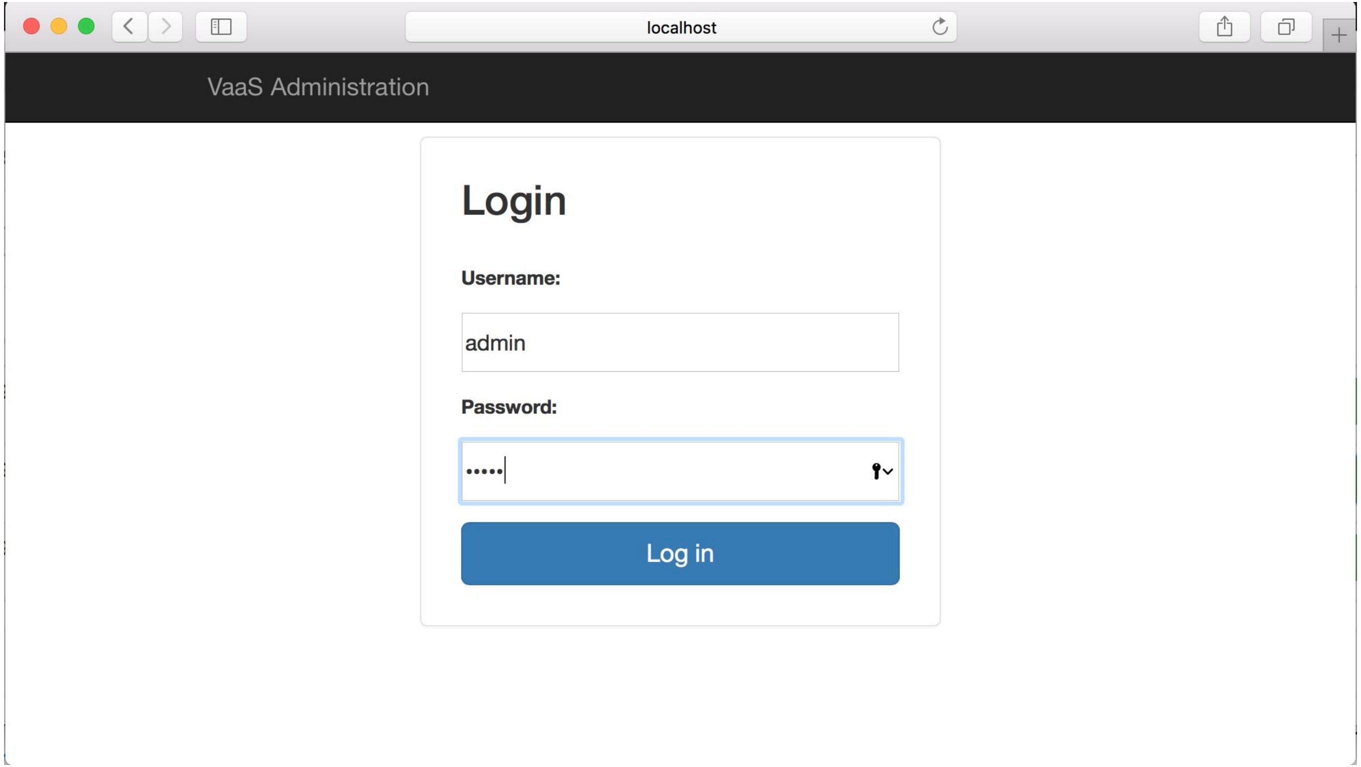
VirtualBox

Vagrant

Run development environment

```
git clone https://github.com/allegro/vaas.git
cd vaas
vagrant up
open http://localhost:3030
```

VaaS - contribute :)



localhost

VaaS Administration

Login

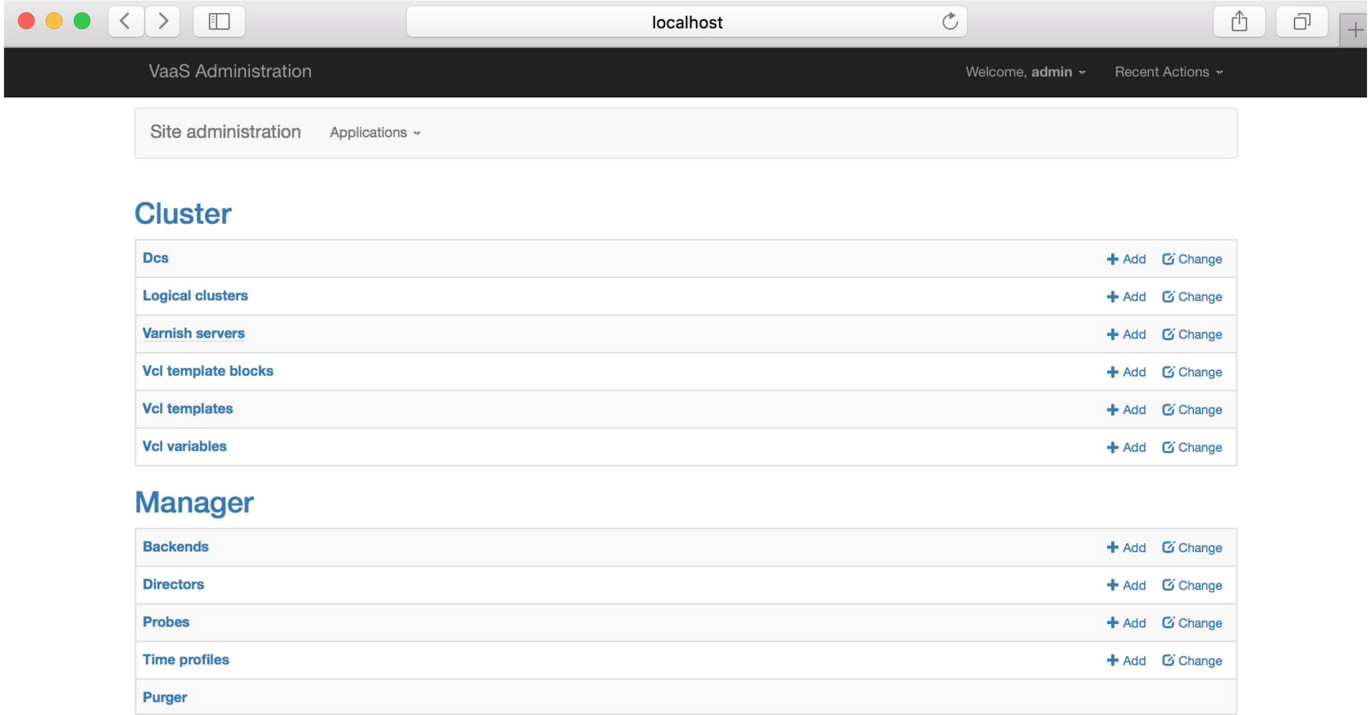
Username:

admin

Password:

•••••

VaaS - contribute :)



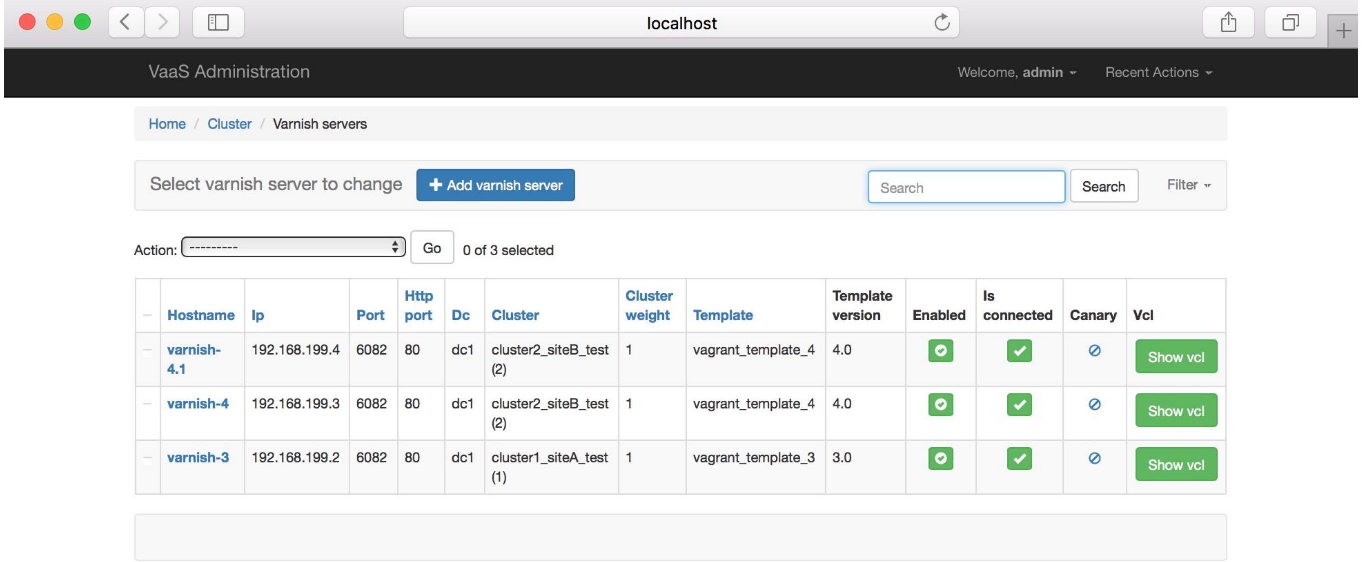
Cluster

Dcs	+ Add	Change
Logical clusters	+ Add	Change
Varnish servers	+ Add	Change
Vcl template blocks	+ Add	Change
Vcl templates	+ Add	Change
Vcl variables	+ Add	Change

Manager

Backends	+ Add	Change
Directors	+ Add	Change
Probes	+ Add	Change
Time profiles	+ Add	Change
Purger		

VaaS - contribute :)



Home / Cluster / Varnish servers

Select varnish server to change

Action: ---------- Go 0 of 3 selected

	Hostname	Ip	Port	Http port	Dc	Cluster	Cluster weight	Template	Template version	Enabled	Is connected	Canary	Vcl
□	varnish-4.1	192.168.199.4	6082	80	dc1	cluster2_siteB_test (2)	1	vagrant_template_4	4.0	✓	✓	⊘	Show vcl
□	varnish-4	192.168.199.3	6082	80	dc1	cluster2_siteB_test (2)	1	vagrant_template_4	4.0	✓	✓	⊘	Show vcl
□	varnish-3	192.168.199.2	6082	80	dc1	cluster1_siteA_test (1)	1	vagrant_template_3	3.0	✓	✓	⊘	Show vcl

VaaS - useful links

1. https://vaas.readthedocs.io/en/latest/
2. https://github.com/allegro/vaas/
3. https://allegro.tech/2015/09/vaas-open-source-management-tool-for-varnish-cache.html

Q&A