Varnish

In this lesson, we'll discuss Varnish.

WE'LL COVER THE FOLLOWING

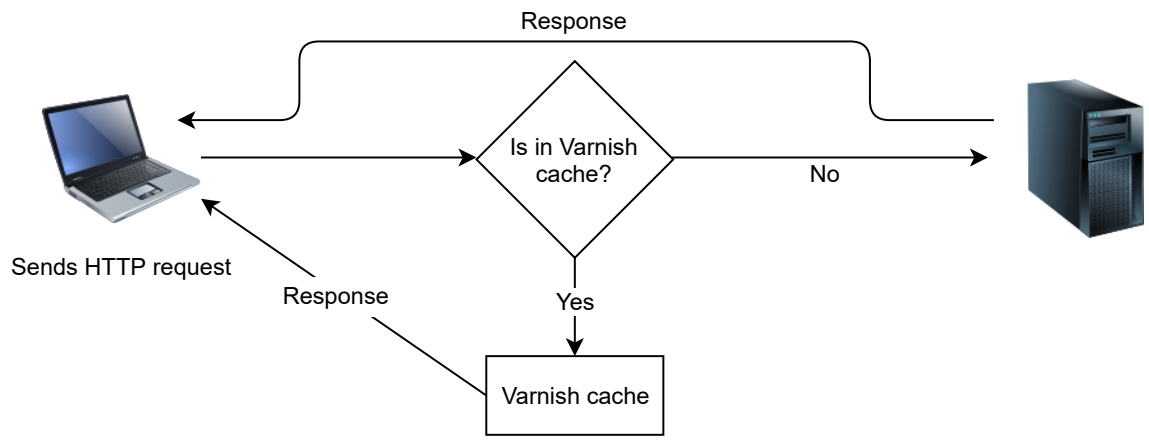
Introduction

Varnish is a web cache and is used as an ESI implementation in the example.

Varnish is mainly used for optimizing web servers:

1. It intercepts the HTTP requests to web servers.
2. It then caches the responses.
3. It forwards only those requests to the web servers that are **not** in the cache.

This improves performance.



Caching with Varnish

Licence and support

Varnish is licensed under a BSD license. The cache is mainly developed by Varnish Software, which also provides commercial support.

Caching with HTTP and HTTP headers

Caching data correctly is not a trivial matter. Above all, the questions are **when can content be retrieved from the cache and when does the data have to be retrieved from the web server** because the data in the cache no longer represents the current state? Varnish uses its own HTTP headers for this.

The HTTP protocol has very good support for caching through its HTTP headers. The control over whether data is cached rests with the web server, which informs the cache of the settings via HTTP headers. Only the web server can decide whether a page can be cached because that depends on the domain logic.

Varnish Docker containers

In the example, Varnish runs in a Docker container which contains an Ubuntu 14.04 LTS. On the Ubuntu image, the Varnish version from the official Varnish repository will be installed.

Varnish configuration

Varnish offers a powerful configuration language. For this example, Varnish is installed in its own Docker container. The configuration can be found in the file `default.vcl` in the directory `docker/varnish/`. Here is an extract with the essential settings.

```
vcl 4.0;

backend default {
    .host = "order";
    .port = "8080";
}

backend common {
    .host = "common";
    .port = "8180";
}

sub vcl_recv vcl_recv {
    if (req.url ~ "^/common") {
       set req.backend_hint = common;
    }
}

sub vcl_backend_response{
    set beresp.do_esi = true;
    set beresp.ttl = 30s;
    set beresp.grace = 15m;
}
```

- `vcl 4.0;` chooses version 4 of the Varnish configuration language.
- The first backend has the name `default`. Each request arriving at the Varnish cache is passed on to the web server if there is no other configuration. The hostname `order` is resolved by Docker Compose. The `default` backend is the *order* microservice, which implements all functions to accept and display orders.
- The second backend has the name `common` and is provided by the host of the same name. In this case, Docker compose resolves the name to a Docker container. The `common` service provides headers and footers for the HTML pages of the microservices and Bootstrap as a shared library for the UI of the microservices.
- When an HTTP request arrives for a URL where the path starts with `/common`, the HTTP request is redirected to the `common` backend. The code

of the subroutine `vcl_recv` is responsible for this. Varnish automatically calls this routine to determine the routes for the requests.

- The subroutine `vcl_backend_response` configures Varnish. The present configuration not only enables ESIs but also implements the functionality of a reverse proxy and redirects requests to specific microservices.
    - `beresp.do_esi` configured Varnish so it interprets ESI tags.
    - `beresp.ttl` turns on the caching. Each page is cached for 30 seconds. However, this caching is very simple: if a new order has been created, it is not displayed until the cache has been invalidated. This can take up to 30 seconds. It would be better if a new order leads to the invalidation of the cache. This is still relatively easy to implement in the example, but in a complex application, it can be difficult to invalidate the correct pages. For example, goods are displayed on product pages and order pages, so several pages need to be invalidated if the data for the goods is changed. Thus, simple time-based caching can be the better solution, being easy to implement and possibly doing a good enough job. There is a chapter about cache invalidation in the Varnish book.
    - Finally, `beresp.grace` ensures that if a backend fails, the web pages are cached for 15 minutes. This can temporarily compensate for a backend failure. Of course, this works only if the web page is already in the cache because it has been accessed before the failure. Therefore, if the page is not in the cache or not cacheable at all, this feature does not help.

Load balancing

You can add load balancing to the Varnish configuration, thereby splitting the requests to the microservices across multiple microservices by defining multiple `backends` in the configuration. However, you can of course also rely on an external load balancer. In that case, Varnish would do only ESI and caching.

Evaluation of VCL

As you can see, VCL is a very powerful language that has many possibilities

for manipulating HTTP requests. That is essential, for example, in a case where you can only be sure a request can be cached if it does not contain cookies as cookies could change the response; therefore, VCL must be able to remove cookies.

A comprehensive documentation of Varnish and VCL can be found in the free Varnish book.

QUIZ

1

What would be a side effect of setting the ttl to 15m, as per the following code snippet, in a highly interactive E-commerce app?

```
sub vcl_backend_response{
    set beresp.do_esi = true;
    set beresp.ttl = 5m;
    set beresp.grace = 15m;
}
```

COMPLETED 0%

1 of 3

The next lesson is about ESI.