IP geolocation / GeoIP with Varnish

Overview

Often we need to determine clients IP location and process some actions before sending request to backend with application.

Varnish itself has no built-in functionality for this task, but can be added with VMOD.

Thanks to Open Source community - GeoIP functionality has been already realized by various people in different GeoIP VMODs.

This post will explain how to build one of the GeoIP VMODs and show few usage examples

Installation

> **Varnish 3**
>
> > **Debian / Ubuntu**
> >
> > Follow installation instructions to setup official repository.
> >
> > Prepare your environment by installing Git and dependency packages.
> >
> > ```
> > apt-get update
> > apt-get install git libvarnishapi-dev libgeoip-dev
> > apt-get build-dep varnish
> > ```
> >
> > Grab Varnish sources
> >
> > ```
> > mkdir ~/src
> > cd ~/src
> > apt-get source varnish
> > ```
> >
> > Change your position to Varnish source folder and build Varnish
> >
> > ```
> > cd varnish-3.0.7/
> > ./autogen.sh
> > ./configure --prefix=/usr
> > make
> >
> > cd ~/src
> > git clone https://github.com/varnish/libvmod-geoip
> > cd ~/src/libvmod-geoip
> > git checkout 3.0
> > ```
> >
> > Last step is compilation & installation of GeoIP VMOD
> >
> > ```
> > ./autogen.sh
> > ./configure VARNISHSRC=~/src/varnish-3.0.7/ VMODDIR=/usr/lib/varnish/vmods
> > make
> > make install
> > ```
> >
> > where `/usr/lib/varnish/vmods` is typical location of Varnish VMODs on Debian-like systems. If you failed here, please re-check your paths and change command correspondingly to them.
>
> > **Centos 6**
> >
> > Follow installation instructions to setup official repository.
> >
> > Prepare your environment by installing Git and dependency packages.
> >
> > ```
> > yum install git varnish-libs-devel GeoIP-devel GeoIP-GeoLite-data automake autoconf libtool ncurses-devel libxslt groff pcre-devel pkgconf
> > ```
> >
> > Grab Varnish sources
> >
> > ```
> > mkdir ~/src
> > cd ~/src
> > git clone https://github.com/varnish/Varnish-Cache.git
> > cd ~/src/Varnish-Cache
> > git checkout varnish-3.0.7
> > ```
> >
> > Build Varnish
> >
> > ```
> > ./autogen.sh
> > ./configure --prefix=/usr
> > make
> >
> > cd ~/src
> > git clone https://github.com/varnish/libvmod-geoip
> > cd ~/src/libvmod-geoip
> > git checkout 3.0
> > ```
> >
> > Last step is compilation & installation of GeoIP VMOD

```
./autogen.sh
./configure VARNISHSRC=~/src/Varnish-Cache/ VMODDIR=/usr/lib64/varnish/vmods/
make
make install
```

where `/usr/lib64/varnish/vmods/` is typical location of Varnish VMODs on RHEL-like systems. If you failed here, please re-check your paths and change command correspondingly to them.

Centos 7

Follow installation instructions to setup official repository.

Grab Varnish sources

```
mkdir ~/src
cd ~/src
git clone https://github.com/varnish/Varnish-Cache.git
cd ~/src/Varnish-Cache
git checkout varnish-3.0.7
```

Build Varnish

```
./autogen.sh
./configure --prefix=/usr
make

cd ~/src
git clone https://github.com/varnish/libvmod-geoip
cd ~/src/libvmod-geoip
git checkout 3.0
```

Last step is compilation & installation of GeoIP VMOD

```
./autogen.sh
./configure VARNISHSRC=~/src/Varnish-Cache/ VMODDIR=/usr/lib64/varnish/vmods/
make
make install
```

where `/usr/lib64/varnish/vmods/` is typical location of Varnish VMODs on RHEL-like systems. If you failed here, please re-check your paths and change command correspondingly to them.

Varnish 4

Debian / Ubuntu

Follow installation instructions to setup official repository.

Prepare your environment by installing Git and dependency packages.

```
apt-get update
apt-get install varnish git libvarnishapi-dev libgeoip-dev
apt-get build-dep varnish
```

Now it is a time to clone GeoIP VMOD source code from GitHub repository

```
cd ~/src
git clone https://github.com/varnish/libvmod-geoip
```

Last step is compilation & installation of GeoIP VMOD

```
cd ~/src/libvmod-geoip
./autogen.sh
./configure
make
make install
```

By default, `/usr/lib/varnish/vmods/` folder used to store Varnish VMODs on Debian-like systems, but you could redefine destination path on configure step with `./configure VMODDIR=/your/path/to/install/vmods`

Centos 6

Follow installation instructions to setup official repository.

Prepare your environment by installing Git and dependency packages.

```
yum install varnish git varnish-libs-devel
yum install GeoIP-devel GeoIP-GeoLite-data automake autoconf libtool ncurses-devel libxslt groff pcre-devel pkgconfig jemalloc-devel pytho
```

Building VMODs for Varnish 4 requires autoconf at least 2.64 version. So, we need to install some recent version manually, as CentOS provides 2.63

```
cd ~/src
wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz
tar xvf autoconf-2.69.tar.gz
cd autoconf-2.69/
./configure
make
make install
```

> Now it is a time to clone GeoIP VMOD source code from GitHub repository
>
> ```
> cd ~/src
> git clone https://github.com/varnish/libvmod-geoip
> cd ~/src/libvmod-geoip
> ```
>
> Last step is compilation & installation of GeoIP VMOD
>
> ```
> ./autogen.sh
> ./configure
> make
> make install
> ```
>
> where `/usr/lib64/varnish/vmods/` is typical location of Varnish VMODs on RHEL-like systems. If you failed here, please re-check your paths and change command correspondingly to them.

> **Centos 7**
>
> Follow installation instructions to setup official repository.
>
> Prepare your environment by installing Git and dependency packages.
>
> ```
> yum install varnish varnish-libs varnish-libs-devel
> yum install git GeoIP-devel GeoIP-update automake autoconf libtool ncurses-devel libxslt groff pcre-devel pkgconfig python-docutils readli
> ```
>
> Now it is a time to clone GeoIP VMOD source code from GitHub repository
>
> ```
> cd ~/src
> git clone https://github.com/varnish/libvmod-geoip
> cd ~/src/libvmod-geoip
> ```
>
> Last step is compilation & installation of GeoIP VMOD
>
> ```
> ./autogen.sh
> ./configure
> make
> make install
> ```
>
> By default, `/usr/lib64/varnish/vmod/` folder used to store Varnish VMODs on RHEL-like systems, but you could redefine destination path on configure step with `./configure VMODDIR=/your/path/to/install/vmods`

VMOD installation could be verified with build of simple VCL

Put this code into `~/src/test.vcl`

```
vcl 4.0;
backend default {
  .host = "127.0.0.1";
  .port = "8080";
}
import geoip;
```

Run command

```
varnishd -C -f ~/src/test.vcl
```

If no error raised, then you successfully installed GeoIP VMOD

Usage

You will be able to use VMOD functions after libvmod-geoip installation, such as determining client country code or country name

Description of functions available on libvmod-geoip GitHub page https://github.com/varnish/libvmod-geoip

You need to import module to be able to use its functions Insert import line to your vcl file before include and subroutines declarations.

```
import geoip;
```

Examples

Getting country code

Add custom header with two letter country code returned by VMOD function

Varnish 3

```
sub vcl_recv {
    ...
    set req.http.X-Country-Code = geoip.client_country_code();
    ...
}
```

Varnish 4

```
sub vcl_recv {
    ...
```

```
    set req.http.X-Country-Code = geoip.country_code("" + client.ip);
    ...
}
```

Storing different cache objects per country

Varnish 3

```
import std;

sub vcl_recv {
    ...
    set req.http.X-Country-Code = geoip.client_country_code();
    ...
}

sub vcl_hash {
    ...
    hash_data(req.http.X-Country-Code);
    ...
}
```

Varnish 4

```
import std;

sub vcl_recv {
    ...
    set req.http.X-Country-Code = geoip.country_code("" + client.ip);
    ...
}

sub vcl_hash {
    ...
    hash_data(req.http.X-Country-Code);
    ...
}
```

Or, for example, separate caches for Europe countries and all other world

Here I'm using regular expression to determine Europe country

```
import std;

sub vcl_hash {
    ...
    if (req.http.X-Country-Code ~ "(AL|AD|AZ|AT|AM|BE|BA|BG|BY|HR|CY|CZ|DK|EE|FO|FI|AX|FR|GE|DE|GI|GR|VA|HU|IS|IE|IT|KZ|LV|LI|LT|LU|MT|MC|MD|
    {
        hash_data("europe");
    }
    ...
}
```

Redirect users to separate two-letter language domain

GeoIP returns two letters only and redirect to domains with more letters will require more complicated logic

Requires import of libvmod-std (built-in Varnish VMOD)

Varnish 3

```
import std;
import geoip;

sub vcl_recv {
...
    # Extract country code
    set req.http.X-Country-Code = geoip.client_country_code();
    # Normalize country code to lower case
    set req.http.X-Country-Code = std.tolower(req.http.X-Country-Code);

    # do redirect for www.example.com host and French, German and Russian people
    if (req.http.host == "www.example.com" && req.http.X-Country-Code ~ "(fr|de|ru)" ) {
        # use dummy error code for our redirect
        error 750 "Moved Temporarily";
    }
...
}

sub vcl_error {
...
    # Describe what Varnish should do with this dummy code
    if (obj.status == 750) {
        # Send location header to browser with new country host
        set obj.http.Location = "http://" + req.http.X-Country-Code + ".example.com";
        set obj.status = 302;
        return(deliver);
    }
...
}
```

Varnish 4

```
import std;
import geoip;

sub vcl_recv {
...
    # Extract country code
    set req.http.X-Country-Code = geoip.country_code("" + client.ip);
    # Normalize country code to lower case
    set req.http.X-Country-Code = std.tolower(req.http.X-Country-Code);

    # do redirect for www.example.com host and French, German and Russian people
    if (req.http.host == "www.example.com" && req.http.X-Country-Code ~ "(fr|de|ru)" ) {
        # use dummy error code for our redirect
        return(synth(750, "Moved Temporarily"));
    }
...
}

sub vcl_synth {
...
    # Describe what Varnish should do with this dummy code
    if (resp.status == 750) {
        # Send location header to browser with new country host
        set resp.http.Location = "http://" + req.http.X-Country-Code + ".example.com";
        set resp.status = 302;
        return(deliver);
    }
...
}
```

Pre-set cookie value

Pre-setting cookie value may be used to defining some default values on the page for users visiting your site first time

Varnish 3

```
import geoip;

sub vcl_recv {
...

    # Check if our magic cookie not present in request
    if (req.http.Cookie !~ "MagicCookie") {
        # Extract country code
        set req.http.X-Country-Code = geoip.client_country_code();

        # Set some value for United Kingdom and Ireland
        if (req.http.X-Country-Code ~ "(GB|IE)") {
            set req.http.Cookie = "MagicCookie=XXX;" + req.http.Cookie;
        }
        # Some other value for Denmark, Finland, Norway and Sweden
        elsif (req.http.X-Country-Code ~ "(DK|FI|NO|SE)") {
            set req.http.Cookie = "MagicCookie=YYY;" + req.http.Cookie;
        }
        # And something completely different for all other world
        else {
            set req.http.Cookie = "MagicCookie=ZZZ;" + req.http.Cookie;
        }
    }

...
}
```

Varnish 4

```
import geoip;

sub vcl_recv {
...

    # Check if our magic cookie not present in request
    if (req.http.Cookie !~ "MagicCookie") {
        # Extract country code
        set req.http.X-Country-Code = geoip.country_code("" + client.ip);

        # Set some value for United Kingdom and Ireland
        if (req.http.X-Country-Code ~ "(GB|IE)") {
            set req.http.Cookie = "MagicCookie=XXX;" + req.http.Cookie;
        }
        # Some other value for Denmark, Finland, Norway and Sweden
        elsif (req.http.X-Country-Code ~ "(DK|FI|NO|SE)") {
            set req.http.Cookie = "MagicCookie=YYY;" + req.http.Cookie;
        }
        # And something completely different for all other world
        else {
            set req.http.Cookie = "MagicCookie=ZZZ;" + req.http.Cookie;
        }
    }

...
}
```

Country codes may be taken from MaxMind site http://dev.maxmind.com/geoip/legacy/codes/iso3166/

If you know some other example, write few words to me and I will add it to article :)

- varnish
- modules
- vmod
- vmods
- geolocation
- geoip