Web Applications

- BookStack on CentOS 7
- Evergreen ILS on Ubuntu 18.04
- Matomo on CentOS 7
- mod_GeoIP on CentOS 7

BookStack on CentOS 7

BookStack is a simple, self-hosted, easy-to-use platform for sharing and storing information.

What you need:

Fresh install of CentOS 7 or other RHEL7 clone.

EPEL and IUS Community Project repositories.

Add the repositories

`yum -y install epel-release`

`yum -y install https://centos7.iuscommunity.org/ius-release.rpm`

Update the system and install the packages

`yum update -y && reboot`

`yum -y install git mariadb101u-server nginx php72u php72u-cli php72u-fpm php72u-gd php72u-json php72u-mbstring php72u-mysqlnd php72u-openssl php72u-tidy php72u-tokenizer php72u-xml php72u-ldap`

Start and secure MySQL

```
systemctl restart mariadb.service   # Start MySQL service
mysql_secure_installation   # Set root password
mysql -u root -p   # Enter root password
```

Create database and user

```
CREATE DATABASE IF NOT EXISTS bookstackdb DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON bookstackdb.* TO 'bookstackuser'@'localhost' IDENTIFIED BY
'YourAwesomePassword' WITH GRANT OPTION;
FLUSH PRIVILEGES;
quit
```

Configure Nginx

Update SOCKS permissions for php-fpm

Update `/etc/php-fpm.d/www.conf` configuration. Look for and update the following settings.

```
listen = /var/run/php-fpm.sock
listen.owner = nginx ; SOCKS permission
listen.group = nginx ; SOCKS permission
listen.mode = 0660 ; SOCKS permission
user = nginx ; PHP-FPM running user
group = nginx ; PHP-FPM running group
php_value[session.save_path]    = /var/www/sessions
```

Backup original Nginx configuration file

`mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.orig\`

Create a new config file

`vim /etc/nginx/nginx.conf`

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
```

```
    default_type        application/octet-stream;

    include /etc/nginx/conf.d/*.conf;
}
```

Bookstack configuration

`vim /etc/nginx/conf.d/bookstack.conf`

```
server {
  listen 80;
  server_name localhost;
  root /var/www/BookStack/public;

  access_log  /var/log/nginx/bookstack_access.log;
  error_log  /var/log/nginx/bookstack_error.log;

  client_max_body_size 1G;
  fastcgi_buffers 64 4K;

  index  index.php;

  location / {
    try_files $uri $uri/ /index.php?$query_string;
  }

  location ~ ^/(?:\.htaccess|data|config|db_structure\.xml|README) {
    deny all;
  }

  location ~ \.php(?:$|/) {
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_pass unix:/var/run/php-fpm.sock;
  }

  location ~* \.(?:jpg|jpeg|gif|bmp|ico|png|css|js|swf)$ {
    expires 30d;
```

```
    access_log off;
  }
}
```

Setting up composer

```
cd /usr/local/bin  # Enter the directory where composer will be installed
curl -sS https://getcomposer.org/installer | php  # Install composer
mv composer.phar composer  # Rename composer
```

Download BookStack code

```
cd /var/www  # Change to where BookStack will be installed
mkdir /var/www/sessions  # Create php sessions directory
git clone https://github.com/BookStackApp/BookStack.git --branch release --single-branch  # Clone the latest from the release branch
cd BookStack && composer install  # Change to the BookStack directory, and let composer do the rest
```

Create the .env file
Update the database settings. The rest of the parameters are safe defaults. A sample is available here:

```
cp .env.example .env  # Copy the example config
vim .env  # Update the new config with database.
```

Set permissions and generate the database. You should still be in the BookStack directory.

```
php artisan key:generate --force  # Generate and update APP_KEY
chown -R nginx:nginx /var/www/{BookStack,sessions}  # Change ownership to the webserver
php artisan migrate --force  # Generate database tables
```

Setup Let's Encrypt

```
yum install -y certbot-nginx # Install certbot
certbot --nginx -d books.clusterapps.com # Run certbot. Follow the prompts.
```

Final cleanup

```
firewall-cmd --permanent --add-service={http,https}
systemctl enable nginx.service mariadb.service php-fpm.service
systemctl reboot
```

Once the system has finished booting, open a browser and head to https://your.url

Evergreen ILS on Ubuntu 18.04

Evergreen is highly-scalable software for libraries that helps library patrons find library materials, and helps libraries manage, catalog, and circulate those materials, no matter how large or complex the libraries.

Evergreen is open source software, licensed under the GNU GPL, version 2 or later.

Learn more here. This guide will be on an Ubuntu 18.04 server.

Some reference for the installation

- The **user** Linux account is the account that you use to log onto the Linux system as a regular user.
- The **root** Linux account is an account that has system administrator privileges. Look for # on the console. Sometimes `sudo` will be used to run single commands as the root user.
- The **opensrf** Linux account is an account that you will create as part of installing OpenSRF.
- The minimum supported version of OpenSRF is 3.0.0.
- **PostgreSQL** is required The minimum supported version is 9.4.
- The **postgres** Linux account is created automatically when you install the PostgreSQL database server.
- The **evergreen** PostgreSQL account is a superuser that you will create to connect to the database server.
- The **egadmin** Evergreen account is an administrator account for Evergreen.
-
    -

OpenSRF

First download and unpack the OpenSRF source as the Linux **user**.

```
wget https://evergreen-ils.org/downloads/opensrf-3.1.0.tar.gz
tar -xvf opensrf-3.1.0.tar.gz
cd opensrf-3.1.0/
```

Issue the following commands from the opensrf folder as the **root** Linux account to install prerequisites.

```
sudo apt-get install make
sudo make -f src/extras/Makefile.install ubuntu-bionic
```

As the Linux **user,** use the `configure` command to configure OpenSRF, and the `make` command to build OpenSRF. The default installation prefix (PREFIX) for OpenSRF is `/opensrf/`.

```
./configure --prefix=/openils --sysconfdir=/openils/conf
make
```

If there were no build errors, then as **root** run the make install.

```
sudo make install
```

Create the **opensrf** user as the **root** user.

```
sudo useradd -m -s /bin/bash opensrf
sudo su -c 'echo "export PATH=\$PATH:/openils/bin" >> /home/opensrf/.bashrc'
sudo passwd opensrf
sudo chown -R opensrf:opensrf /openils
```

Define your public and private OpenSRF domains.
This is a single server setup so the /etc/hosts file will be used to add the public and private addresses.
Use the **root** user to add the following to /etc/hosts file.

```
127.0.1.2       public.localhost        public
127.0.1.3       private.localhost       private
```

Adjusting the system dynamic library path.

```
sudo su -c 'echo /openils/lib > /etc/ld.so.conf.d/opensrf.conf'
sudo ldconfig
```

OpenSRF requires an XMPP (Jabber) server. The ejabberd packages will be used.
First stop the existing service.

```
sudo systemctl stop ejabberd.service
```

Open `/etc/ejabberd/ejabberd.yml` and make the following changes as the **root** user:

> If you don't want to edit this file, download a preconfigured one fromhere

1. Define your public and private domains in the `hosts` directive. For example:
   ```
   hosts:
     - "localhost"
     - "private.localhost"
     - "public.localhost"
   ```
2. Change `starttls_required` to false
3. Change `auth_password_format` to plain
4. Change `shaper:` `normal` and `fast` values to 500000
5. Increase the `max_user_sessions:` `all:` value to 10000
6. Comment out the `mod_offline` directive
   ```
   ##mod_offline:
   ##access_max_user_messages: max_user_offline_messages
   ```
7. Uncomment the `mod_legacy_auth` directive

Or use the downloaded file.

```
sudo mv /etc/ejabberd/{ejabberd.yml,ejabberd.yml.org}
sudo wget -O /etc/ejabberd/ejabberd.yml https://git.clusterapps.com/snippets/6/raw
```

Start the ejabberd server:

```
sudo systemctl start ejabberd.service
```

On each domain, you need two Jabber users to manage the OpenSRF communications:

- a `router` user, to whom all requests to connect to an OpenSRF service will be routed; this Jabber user must be named `router`
- an `opensrf` user, which clients use to connect to OpenSRF services; this user can be named anything.

```
sudo ejabberdctl register router private.localhost UseARealPasswordNotthisone
sudo ejabberdctl register opensrf private.localhost UseARealPasswordNotthisone
sudo ejabberdctl register router public.localhost UseARealPasswordNotthisone
sudo ejabberdctl register opensrf public.localhost UseARealPasswordNotthisone
```

As the **opensrf** Linux account, copy the example configuration files to create your locally OpenSRF configuration.

```
sudo su - opensrf
cd /openils/conf
cp opensrf_core.xml.example opensrf_core.xml
cp opensrf.xml.example opensrf.xml
```

Edit `opensrf_core.xml` file to update the username / password pairs to match the Jabber user accounts you created.

1. `<config><opensrf>` = use the private Jabber `opensrf` user
2. `<config><gateway>` = use the public Jabber `opensrf` user
3. `<config><routers><router>` = use the public Jabber `router` user
4. `<config><routers><router>` = use the private Jabber `router` user

Enable the opensrf Linux account to use `srfsh` as the **opensrf** user.

```
cp /openils/conf/srfsh.xml.example /home/opensrf/.srfsh.xml
```

Update the password to match the password you set for the Jabber `opensrf` user at the `private.localhost` domain.

```
vim /home/opensrf/.srfsh.xml
```

Start all OpenSRF services as the **opensrf** Linux account.

```
osrf_control --localhost --start-all
```

1. Start the `srfsh` interactive OpenSRF shell by issuing the following command as the **opensrf** Linux account:
   Starting the `srfsh` interactive OpenSRF shell
   ```
   srfsh
   ```
2. Issue the following request to test the `opensrf.math` service:
   ```
   srfsh# request opensrf.math add 2,2
   ```
   You should receive the value `4`.

Install WebSockets as the **root** user.

```
apt-get install git-core
cd /tmp
git clone https://github.com/disconnect/apache-websocket
cd apache-websocket
apxs2 -i -a -c mod_websocket.c
```

Create the websocket Apache instance

```
sh /usr/share/doc/apache2/examples/setup-instance websockets
```

Remove from the main apache instance

```
a2dismod websocket
```

Change to the directory into which you unpacked OpenSRF, then copy config files.

```
cp examples/apache_24/websockets/apache2.conf /etc/apache2-websockets/
```

Add configuration variables to the end of /etc/apache2-websockets/envvars.

```
export OSRF_WEBSOCKET_IDLE_TIMEOUT=120
export OSRF_WEBSOCKET_IDLE_CHECK_INTERVAL=5
export OSRF_WEBSOCKET_CONFIG_FILE=/openils/conf/opensrf_core.xml
export OSRF_WEBSOCKET_CONFIG_CTXT=gateway
export OSRF_WEBSOCKET_MAX_REQUEST_WAIT_TIME=600
```

Before you can start websockets, you must install a valid SSL certificate in `/etc/apache2/ssl/`.

Evergreen application

Download and extract the archive as the Linux **user.**

```
wget https://evergreen-ils.org/downloads/Evergreen-ILS-3.3.0.tar.gz
tar -xvf Evergreen-ILS-3.3.0.tar.gz
cd Evergreen-ILS-3.3.0/
```

Issue the following commands as the **root** Linux account to install prerequisites using the `Makefile.install` prerequisite installer

```
make -f Open-ILS/src/extras/Makefile.install ubuntu-bionic
```

From the Evergreen source directory, issue the following commands as the **user** Linux account to configure and build Evergreen:

```
PATH=/openils/bin:$PATH ./configure --prefix=/openils --sysconfdir=/openils/conf
make
```

Issue the following command as the **root** Linux account to install Evergreen

```
sudo make install
```

Change ownership of the Evergreen files

```
sudo chown -R opensrf:opensrf /openils
```

Run ldconfig as the **root** Linux account.

```
sudo ldconfig
```

Issue the following commands as the **root** Linux account to copy the Apache configuration files from the Evergreen source archive.

```
sudo cp Open-ILS/examples/apache_24/eg_24.conf /etc/apache2/sites-available/eg.conf
sudo cp Open-ILS/examples/apache_24/eg_vhost_24.conf /etc/apache2/eg_vhost.conf
sudo cp Open-ILS/examples/apache_24/eg_startup /etc/apache2/
```

As the **root** Linux account, edit the `eg.conf` file that you copied.

To enable access to the offline upload / execute interface from any workstation on any network, make the following change (and note that you **must** secure this for a production instance):

- Replace `Require host 10.0.0.0/8` with `Require all granted`

As the **root** Linux account, edit `/etc/apache2/envvars`. Change `export APACHE_RUN_USER=www-data` to `export APACHE_RUN_USER=opensrf`.

As **root** edit `/etc/apache2/apache2.conf` and make the following changes.

```
sudo vim /etc/apache2/apache2.conf
```

1. Change `KeepAliveTimeout` to `1`.
2. Change `MaxKeepAliveRequests` to `100`.

As the **root** Linux account, configure the prefork module to start and keep enough Apache servers available to provide quick responses to clients.

```
sudo vim /etc/apache2/mods-available/mpm_prefork.conf
```

```
<IfModule mpm_prefork_module>
  StartServers          15
  MinSpareServers        5
  MaxSpareServers       15
```

```
    MaxRequestWorkers       75
    MaxConnectionsPerChild 500
</IfModule>
```

As the **root** user, enable the mpm_prefork module:

```
sudo a2dismod mpm_event
sudo a2enmod mpm_prefork
```

As the **root** Linux account, enable the Evergreen site:

```
sudo a2dissite 000-default
sudo a2ensite eg.conf
```

As the **root** Linux account, enable Apache to write to the lock directory

```
sudo chown opensrf /var/lock/apache2
```

To configure OpenSRF for Evergreen, issue the following commands as the **opensrf** Linux account. (Yes you did edit these files earlier but now there are more settings)

```
cp -b /openils/conf/opensrf_core.xml.example /openils/conf/opensrf_core.xml
cp -b /openils/conf/opensrf.xml.example /openils/conf/opensrf.xml
```

Edited the `opensrf_core.xml` as the **opensrf** user and enter the same passwords from the OpenSRF install steps.

To enable the default set of hooks, issue the following command as the **opensrf** Linux account:

```
cp -b /openils/conf/action_trigger_filters.json.example /openils/conf/action_trigger_filters.json
```

Evergreen database

In production environments, this would be on a dedicated server or cluster. For this deployment the database will be on the same server. If you are serious about your setup you would never install the application and the database on the same server. This is for testing only. For the best performance, run PostgreSQL on RHEL or Solaris.

Installing PostgreSQL server packages as **root**. Use the Makefile provided in the Evergreen archive.

```
sudo make -f Open-ILS/src/extras/Makefile.install postgres-server-ubuntu-bionic
```

Issue the following command as the **postgres** Linux account to create a new PostgreSQL superuser named `evergreen`.

```
sudo su - postgres
createuser -s -P evergreen
```

Issue the following command as the **root** Linux account from inside the Evergreen source directory, replacing <user>, <password>, <hostname>, <port>, and <dbname> with the appropriate values for your PostgreSQL database (where <user> and <password> are for the **evergreen** PostgreSQL account you just created), and replace <admin-user> and <admin-pass> with the values you want for the **egadmin** Evergreen administrator account:

```
perl Open-ILS/src/support-scripts/eg_db_config --update-config \
 --service all --create-database --create-schema --create-offline \
 --user <user> --password <password> --hostname <hostname> --port <port> \
 --database <dbname> --admin-user <admin-user> --admin-pass <admin-pass>
```

If you add the `--load-all-sample` parameter to the `eg_db_config` command, a set of authority and bibliographic records, call numbers, copies, staff and regular users, and transactions will be loaded into your target database. This sample dataset is commonly referred to as the *concerto* sample data, and can be useful for testing out Evergreen functionality and for creating problem reports that developers can easily recreate with their own copy of the *concerto* sample data.

Starting Evergreen

Start the `memcached` and `ejabberd` services

```
sudo systemctl enable --now ejabberd
sudo systemctl enable --now memcached
```

As the **opensrf** Linux account, start Evergreen. The `-l` flag in the following command is only necessary if you want to force Evergreen to treat the hostname as `localhost`;

```
osrf_control -l --start-all
```

if you configured `opensrf.xml` using the real hostname of your machine as returned by `perl -ENet::Domain 'print Net::Domain::hostfqdn() . "\n";'`, you should not use the `-l` flag. In a multi server setup, do not use localhost.

As the **opensrf** Linux account, generate the Web files needed by the web staff client and catalogue and update the organization unit proximity. Do this the first time you start Evergreen, and after that each time you change the library org unit configuration.

```
autogen.sh
```

As the **root** Linux account, restart the Apache Web server:

```
systemctl restart apache2
```

Testing connections

Once you have installed and started Evergreen, test your connection to Evergreen via `srfsh`.
As the **opensrf** Linux account. <admin-user> <admin-pass> are the egadmin username and password created earlier.

```
/openils/bin/srfsh
srfsh% login <admin-user> <admin-pass>
```

The output should look like this:

```
##############################################
Received Data: "$2a$10$londKQogYvvF71H92Wwpme"

------------------------------------
Request Completed Successfully
Request Time in seconds: 0.052501
------------------------------------

Received Data: {
"ilsevent":0,
"textcode":"SUCCESS",
"desc":"Success",
"pid":32474,
"stacktrace":"oils_auth.c:636",
"payload":{
"authtoken":"bd4f67a646ee4c39e923a272dc6c79a3",
"authtime":420
}
}

------------------------------------
Request Completed Successfully
Request Time in seconds: 0.171812
------------------------------------
Login Session: bd4f67a646ee4c39e923a272dc6c79a3. Session timeout: 420.000000

##############################################
```

Matomo on CentOS 7

Installing Matomo

Matomo, formerly known as Piwik, is an open source web analytics application. It rivals Google Analytics and includes even more features and allows you to brand your brand and send out custom daily, weekly, and monthly reports to your clients.

First let’s start by ensuring your system is up-to-date and has the needed repositories.

```
yum -y install epel-release
yum -y install https://centos7.iuscommunity.org/ius-release.rpm
yum clean all
yum -y update
reboot # if kernel updated
```

Install needed packages

```
yum -y install wget mariadb mariadb-server mysql httpd openssl mod_ssl php72u-json mod_php72u php72u-gd php72u-imap php72u-ldap php72u-odbc pear1u php72u-xml php72u-xmlrpc php72u-mbstring php72u-mysqlnd php72u-snmp php72u-soap php72u-tidy curl curl-devel mcrypt
```

Configure MySQL

```
systemctl restart mariadb.service   # Start MySQL service
mysql_secure_installation   # Set root password
mysql -u root -p   # Enter root password
```

Add the database and user.

```
CREATE DATABASE IF NOT EXISTS matomodb DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON matomodb.* TO 'matomouser'@'localhost' IDENTIFIED BY 'YourAwesomePassword'
WITH GRANT OPTION;
FLUSH PRIVILEGES;
quit
```

Installing Matomo on CentOS 7.

```
cd /var/www
wget https://builds.matomo.org/piwik.zip
```

Unpack the Matomo archive to the document root directory on your server.

```
unzip piwik.zip -d /var/www/html/
mv /var/www/html/piwik/ /var/www/html/matomo/
```

Update owner on Matomo files and folders

`chown -R apache:apache /var/www/html/matomo`

Configure Apache
Create Apache virtual host for Matomo . First create ‘/etc/httpd/conf.d/vhosts.conf’ file

`vim /etc/httpd/conf.d/vhosts.conf`

```
IncludeOptional vhosts.d/*.conf
```

Create the virtual host.

```
mkdir /etc/httpd/vhosts.d/
vim /etc/httpd/vhosts.d/yourdomain.com.conf
```

Add the following to the new vhost config.

```
<VirtualHost YOUR_SERVER_IP:80>
ServerAdmin webmaster@yourdomain.com
DocumentRoot /var/www/html/matomo
ServerName yourdomain.com
ServerAlias www.yourdomain.com
ErrorLog "/var/log/httpd/yourdomain.com-error_log"
CustomLog "/var/log/httpd/yourdomain.com-access_log" combined

<Directory "/var/www/html/matomo/">
DirectoryIndex index.html index.php
Options FollowSymLinks
AllowOverride All
Require all granted
</Directory>
</VirtualHost>
```

Start Apache

```
systemctl start httpd
```

Verify that Apache is running by checking the status of the service:

```
systemctl status httpd
```

Install certbot to handle SSL

```
yum install -y mod_ssl python-certbot-apache
```

Run Certbot to secure the Apache site

```
certbot --apache -d site.example.com
```

Enable services and reboot the server and make sure it works.

```
systemctl enable httpd mariadb
reboot
```

Browse to https://your.sitename.com and follow the Matomo setup steps.

For details see https://matomo.org/docs/installation/

Installing libmaxminddb

Install git and PHP development libraries.

```
yum -y install php72u-devel git automake autoconf libtool
```

To install the library you need to download it's latest tar ball and extract it, or clone their git repository

```
git clone --recursive https://github.com/maxmind/libmaxminddb
```

When cloning from git, run `./bootstrap` from the libmaxminddb directory and then run the commands.

```
./configure
make
sudo make install
sudo ldconfig
```

You can find more details about installing the library in their README

Installing Extension

After successfully installing libmaxmindb, you need to download or checkout MaxMind-DB-Reader-php.
Then run the following commands from the top-level directory of this distribution:

```
cd ext
phpize
./configure
make
sudo make install
```

You then must load your extension. The recommend method is to add the following to your php.ini file:

```
extension=maxminddb.so
```

Now restart the webserver and the GeoIP 2 PHP provider should mention if the extension is loaded in Matomo (Piwik) > Settings > Geolocation.

Note: You may need to install the PHP development package on your OS such as php5-dev for Debian-based systems or php-devel for RedHat/Fedora-based ones.

If after installing, you receive an error that `libmaxminddb.so.0` is missing you may need to add the `lib` directory in your `prefix` to your library path. On most Linux distributions when using the default prefix (`/usr/local`), you can do this by running the following commands:

```
echo /usr/local/lib  >> /etc/ld.so.conf.d/local.conf
ldconfig
```

Download the GeoIP database and copy it to Matomo's path/to/matomo/misc/ subdirectory.

```
wget https://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz
cp GeoLite2-City_20190312/GeoLite2-City.mmdb /path/to/matomo/misc/
```

mod_GeoIP on CentOS 7

Mod_GeoIP is an Apache module that can be used to get the geographic location of IP address of the visitor into the Apache web server. The module allows you to determine the visitor's country and location. It is specially useful for Geo Ad Serving, Target Content, Spam Fighting, Fraud Detection, Redirecting/Blocking visitors based on their country and much more.

GeoIP module allows system administrators to redirect or block web traffic according on the client geographical location. The geographical location is learned via client IP address.

Mod_GeoIP has two versions, one is Free and another one is Paid.

Enable EPEL Repository

Mod_Geoip is not available under official repository, install and enable third party EPEL repository.

```
yum install epel-release
```

Install Mod_GeoIP

Once you've **EPEL** repository enabled on your system, you can simple install **mod_geoip** by running following command with their dependency packages.

```
yum install mod_geoip GeoIP GeoIP-devel GeoIP-data zlib-devel
```

Download latest Geo Databases

It's good idea to download latest **Geo City** and **Country Database** to stay updated.

```
cd /usr/share/GeoIP/
mv GeoIP.dat GeoIP.dat_org
wget http://geolite.maxmind.com/download/geoip/database/GeoLite2-Country.tar.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz
gunzip GeoLite2-Country.tar.gz
gunzip GeoLite2-City.tar.gz
```

Enable Mod_GeoIP in Apache

After the module has been installed, open and edit the module main configuration file, with a command line text editor such as **vim**, and activate the module server-wide, as illustrated in the

below excerpt.

```
vim /etc/httpd/conf.d/geoip.conf
```

Set the line GeoIPEnable from **Off** to **On**. Also, make sure you add the absolute path to GeoIP database file.

```
<IfModule mod_geoip.c>
GeoIPEnable On
GeoIPDBFile /usr/share/GeoIP/GeoIP.dat MemoryCache
</IfModule>
```

Restart the **Apache** service to reflect changes.

```
systemctl restart httpd
```

If you are running multiple sites, It's not recommended to turn on GeoIP module server-wide. You should enable the GeoIP module only in `<Location>` or `<Directory>` blocks where you would actually perform the traffic redirection or block.

Updating GeoIP Database

GeoIP database is updated beginning of every month. So, its is very important to keep GeoIP database up-to-date. To download latest version of database use the following command.

```
cd /usr/share/GeoIP/
mv GeoIP.dat GeoIP.dat_org
wget http://geolite.maxmind.com/download/geoip/database/GeoLite2-Country.tar.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz
gunzip GeoLite2-Country.tar.gz
gunzip GeoLite2-City.tar.gz
```

For more information about mod_geoip and its usage can be found at http://www.maxmind.com/app/mod_geoip.