Switch2OSM

Take back control of your maps

Manually building a tile server (Debian 11)

Using Tiles

Getting started with Leaflet

Getting started with OpenLayers

Serving Tiles

Manually building a tile server (Debian 11)

Manually building a tile server (Ubuntu 22.04 LTS)

Manually building a tile server (Ubuntu 20.04 LTS)

Using a Docker container

Manually building a tile server (Ubuntu 18.04 LTS)

Monitoring using Munin

Updating your database as people edit OpenStreetMap (osm2pgsql-replication)

Updating your database as people edit OpenStreetMap (PyOsmium)

Updating your database as people edit OpenStreetMap (Osmosis)

Linux 11 testing (bullseye), and were tested in November 2020.

Software installation

The OSM tile server stack is a collection of programs and libraries that work together to create a tile server. As so often with OpenStreetMap, there are many ways to achieve this goal and nearly all of the components have alternatives that have various specific advantages and disadvantages. This tutorial describes the most standard version that is similar to that used on the main OpenStreetMap.org tile servers.

It consists of 5 main components: mod_tile, renderd, mapnik, osm2pgsql and a postgresql/postgis database. Mod_tile is an apache module that serves cached tiles and decides which tiles need re-rendering - either because they are not yet cached or because they are outdated. Renderd provides a priority queueing system for different sorts of requests to manage and smooth out the load from rendering requests. Mapnik is the software library that does the actual rendering and is used by renderd.

Thanks to the work done by the Debian maintainers to incorporate the latest versions of these packages into Debian 11, these instructions are somewhat shorter than previous versions.

These instructions are have been written and tested against a newly-installed Debian 11 Testing server. Debian 11 hasn't been released yet, so of course things may change. If you have got other versions of some software already installed (perhaps you upgraded from an earlier version, or you set up some PPAs to load from) then you may need to make some adjustments.

In order to build these components, a variety of dependencies need to be installed first. Debian doesn't come with "sudo" by default, so we'll need to log on as root to do the first part:

While still logged on as root we'll ensure that the main user account that we are using can "sudo" to root. You'll want to change "youruseraccount" to whatever account you are using in the line below. Don't try and do everything below as root; it won't work.

```
usermod -aG sudo youruseraccount

exit
```

That returns to your user account. Logoff and logon again, and then:

```
sudo whoami
```

That should return "root". At this point, a couple of new accounts have been added. You can see them with "tail /etc/passwd". "postgres" is used for managing the databases that we use to hold data for rendering. "_renderd" is used for the renderd daemon, and we'll need to make sure lots of the commands below are run as that user.

Now you need to create a postgis database. The defaults of various programs assume the database is called gis and we will use the same convention in this tutorial, although this is not necessary. Note that "_renderd" below matches the user that the renderd daemon will run from.

```
sudo -u postgres -i

createuser _renderd

createdb -E UTF8 -O _renderd gis
```

While still working as the "postgres" user, set up PostGIS on the PostgreSQL database:

```
psql
```

(that'll put you at a "postgres=#" prompt)

(it'll answer "You are now connected to database 'gis' as user 'postgres'".)

```
CREATE EXTENSION postgis;
```

(it'll answer CREATE EXTENSION)

```
CREATE EXTENSION hstore;
```

(it'll answer CREATE EXTENSION)

```
ALTER TABLE geometry_columns OWNER TO _renderd;
```

(it'll answer ALTER TABLE)

```
ALTER TABLE spatial_ref_sys OWNER TO _renderd;
```

(it'll answer ALTER TABLE)

```
\q
```

(it'll exit psql and go back to a normal Linux prompt)

```
exit
```

(to exit back to be the user that we were before we did "sudo -u postgres -i" above)

Mapnik

Mapnik was installed above. We'll check that it has been installed correctly by doing this:

```
>>>
```

If python replies with the second chevron prompt »> and without errors, then Mapnik library was found by Python. Congratulations! You can leave Python with this command:

```
>>> quit()
```

Stylesheet configuration

Now that all of the necessary software is installed, you will need to download and configure a stylesheet.

The style we'll use here is the one that use by the "standard" map on the openstreetmap.org website. It's chosen because it's well documented, and should work anywhere in the world (including in places with non-latin placenames). There are a couple of downsides though - it's very much a compromise designed to work globally, and it's quite complicated to understand and modify, should you need to do that.

The home of "OpenStreetMap Carto" on the web is https://github.com/gravitystorm/openstreetmap-carto/ and it has it's own installation instructions at https://github.com/gravitystorm/openstreetmap-carto/blob/master/INSTALL.md , although we'll cover everything that needs to be done here.

Here we're assuming that we're storing the stylesheet details in a directory below "src" below the home directory of the whichever non-root account you are using:

```
mkdir ~/src

cd ~/src

git clone https://github.com/gravitystorm/openstreetmap-carto

cd openstreetmap-carto
```

```
carto -v
```

That should respond with a number that is at least as high as:

```
1.2.0
```

Then we convert the carto project into something that Mapnik can understand:

```
carto project.mml > mapnik.xml
```

You now have a Mapnik XML stylesheet at /home/youraccountname /src/openstreetmap-carto/mapnik.xml .

Loading data

Initially, we'll load only a small amount of test data. Other download locations are available, but "download.geofabrik.de" has a wide range of options. In this example we'll download the data for Azerbaijan, which is currently about 32Mb.

Browse to https://download.geofabrik.de/asia/azerbaijan.html and note the "This file was last modified" date (e.g. "2020-11-13T21:42:03Z"). We'll need that later if we want to update the database with people's susbsequent changes to OpenStreetMap. Download it as follows:

```
mkdir ~/data

cd ~/data

wget https://download.geofabrik.de/asia/azerbaijan-latest.osm.pbf
```

The following command will insert the OpenStreetMap data you downloaded earlier into the database. This step is very disk I/O intensive; importing the full planet might take this process.

```
sudo -u _renderd osm2pgsql -d gis --create --slim  -G --hstore --tag-transform-script ~/src/opens
```

It’s worth explaining a little bit about what those options mean:

```
-d gis
```

The database to work with (“gis” used to be the default; now it must be specified).

```
--create
```

Load data into an empty database rather than trying to append to an existing one.

```
--slim
```

osm2pgsql can use different table layouts; “slim” tables works for rendering.

```
-G
```

Determines how multipolygons are processed.

```
--hstore
```

Allows tags for which there are no explicit database columns to be used for rendering.

```
--tag-transform-script
```

Defines the lua script used for tag processing. This an easy is a way to process OSM tags

Allocate 2.5 Gb of memory to osm2pgsql to the import process. If you have less memory you could try a smaller number, and if the import process is killed because it runs out of memory you'll need to try a smaller number or a smaller OSM extract..

```
--number-processes 1
```

Use 1 CPU. If you have more cores available you can use more.

```
-S
```

Create the database columns in this file (actually these are unchanged from "openstreetmap-carto")

The final argument is the data file to load.

That command will complete with something like "Osm2pgsql took 238s overall".

Creating indexes

Since version v5.3.0, some extra indexes now need to be applied manually:

```
cd ~/src/openstreetmap-carto/

sudo -u _renderd psql -d gis -f indexes.sql
```

It should respond with "CREATE INDEX" 14 times.

Shapefile download

Although most of the data used to create the map is directly from the OpenStreetMap data file that you downloaded above, some shapefiles for things like low-zoom country

```
cd ~/src/openstreetmap-carto/

mkdir data

sudo chown _renderd data

sudo -u _renderd scripts/get-external-data.py
```

This process involves a sizable download and may take some time - not much will appear on the screen when it is running. Some data will go directly into the database, and some will go into a “data” directory below “openstreetmap-carto”. If there is a problem here then the Natural Earth data may have moved - look at this issue and other issues at Natural Earth for more details. If you need to change the Natural Earth download location your copy of this file is the one to edit.

Fonts

In version v5.6.0 and above of Carto, fonts need to be installed manually:

```
cd ~/src/openstreetmap-carto/

scripts/get-fonts.sh
```

Our test data area (Azerbaijan) was chosen both because it was a small area and because some place names in that region have names containing non-latin characters.

Setting up your webserver

Configure renderd

The config file for “renderd” on Debian is “/etc/renderd.conf”. Edit that with a text editor such as nano:

Add a section like the following at the end:

```
[s2o]

URI=/hot/

TILEDIR=/var/lib/mod_tile

XML=/home/accountname/src/openstreetmap-carto/mapnik.xml

HOST=localhost

TILESIZE=256

MAXZOOM=20
```

The location of the XML file "/home/accountname/src/openstreetmap-carto/mapnik.xml" will need to be changed to the actual location on your system. You can change "[s2o]" and "URI=/hot/" as well if you like. If you want to render more than one set of tiles from one server you can - just add another section like "[s2o]" with a different name referring to a different map style. If you want it to refer to a different database to the default "gis" you can, but that's out of the scope of this document. If you've only got 2Gb or so of memory you'll also want to reduce "num_threads" to 2. "URI=/hot/" was chosen so that the tiles generated here can more easily be used in place of the HOT tile layer at OpenStreetMap.org. You can use something else here, but "/hot/" is as good as anything.

When this guide was first written, the version of Mapnik provided by Debian was 3.0, and the "plugins_dir" setting in the "[mapnik]" part of the file was "/usr/lib/mapnik/3.0/input". At the time of writing that's changed to 3.1, and so the relevant value is ""/usr/lib/mapnik/3.1/input". It may change again in the future. If an error occurs when trying to render tiles such as this:

```
An error occurred while loading the map layer 's2o': Could not create datasource for type: 'postg
```

then look in "/usr/lib/mapnik" and see what version directories there are, and also look in "/usr/lib/mapnik/(version)/input" to make sure that a file "postgis.input" exists there.

```
sudo mkdir /var/lib/mod_tile

sudo chown _renderd /var/lib/mod_tile
```

After doing so, restart renderd:

```
sudo /etc/init.d/renderd restart
```

If you look at /var/log/syslog, you should see messages from the “renderd” service. There will initially be some font errors - don’t worry about those for now. Next:

```
sudo /etc/init.d/apache2 restart
```

In syslog you should see a message like:

```
Nov 14 14:24:55 servername apachectl[19119]: [Sat Nov 14 14:24:55.526717 2020] [tile:notice] [pid
```

Next, point a web browser at “http://yourserveripaddress/index.html” (change yourserveripaddress to your actual server address). You should see “Debian Logo Apache2 Debian Default Page”.

If you don’t know what IP address it will have been assigned you can likely use “ifconfig” to find out - if the network configuration is not too complicated it’ll probably be the “inet addr” that is not “127.0.0.1”). If you’re using a server at a hosting provider then it’s likely that your server’s internal address will be different to the external address that has been allocated to you, but that external IP address will have already been sent to you and it’ll probably be the one that you’re accessing the server on currently.

Note that this is just the “http” (port 80) site - you’ll need to do a little bit more Apache configuration if you want to enable https, but that’s out of the scope of these instructions. However, if you use “Let’s Encrypt” to issue certificates then the process of setting that up can also configure the Apache HTTPS site as well.

small map of the world. If you don't, ivestigate the errors that it displays. These will most likely be permissions errors or perhaps related to accidentally missing some steps from the instructions above. If you don't get a tile and get other errors again save the full output in a pastebin and ask a question about the problem somewhere like help.openstreetmap.org.

Viewing tiles

In order to see tiles, we'll cheat and use an html file "sample_leaflet.html" that allows you to view a very simple map. To obtain this:

```
cd /var/www/html

sudo wget https://raw.githubusercontent.com/SomeoneElseOSM/mod_tile/switch2osm/extra/sample_leafl

sudo nano sample_leaflet.html
```

Edit so that the IP address matches yourserveraddress rather than just saying "127.0.0.1". That should allow you to access this server from others. Then browse to "http://yourserveraddress/sample_leaflet.html".

The initial map display will take a little while. You'll be able to zoom in and out, but depending on server speed some tiles may initially display as grey because they can't be rendered in time for the browser. However, once done they'll be ready for the next time that they are needed. If you look in /var/log/syslog you should see requests for tiles.

If desired, you can increase the setting "ModTileMissingRequestTimeout" in "/etc/apache2/conf-available/renderd.conf" from 10 seconds to perhaps 30 or 60, in order to wait longer for tiles to be rendered in the background before a grey tile is given to the user. Make sure you "sudo service renderd restart" and "sudo service apache2 restart" after changing it.

Congratulations. Head over to the using tiles section to create a map that uses your new tile server.