How to Set Up OpenStreetMap Tile Server on Ubuntu 18.04

Last Updated: May 24, 2020 Xiao Guoan (Admin) 82 Comments Ubuntu

OpenStreetMap, aka OSM, is a user-contributed world map that is freely editable. You can think of it as an open-source and self-hosted alternative to Google Maps. This tutorial will show you how to build your own OpenStreetMap tile server on Ubuntu 18.04 so you don't have to use a proprietary map service.

OpenStreetMap Features

- OpenStreetMap data covers the whole world, making it easy to support users in any country or every country.
- OpenStreetMap is updated every minute of every hour of every day, and these updates are available to you in real-time.
- OpenStreetMap data is free and open – there is no subscription fee and no page-view fee.
- OpenStreetMap data is rich and detailed, containing huge amounts of data that is relevant to people on the ground – the people who collected it.

Prerequisites/Hardware Requirements

The required RAM and disk space depend on which country's map you are going to use. For example,

- The UK map requires at least 12G RAM and 60GB disk space.
- The whole planet map requires at least 32G RAM and 1TB SSD disk. It's not viable to use a spinning hard disk for the whole planet map.

You will need more disk space if you are going to pre-render tiles to speed up map loading in the web browser, which is highly recommended. Check this page to see how much disk space are required for pre-rendering tiles. For example, if you are going to pre-render tiles from zoom level 0 to zoom level 15 for the planet map, an extra 460 GB disk space is required.

Another thing to note is that importing large map data, like the whole planet, to PostgreSQL database takes a long time. Consider adding more RAM and especially **using SSD** instead of spinning hard disk to speed up the import process.

If you are going to host the entire world map, I recommend you buy the extra-large VPS from Contabo, which boasts

- A 10 core CPU
- 60 GB RAM
- 1.6 TB Intel Optane SSD

It costs just 26.99 €/month.

Step 1: Upgrade Software

It's always a good practice to update server software before doing any major work on your server. Log into your server via SSH and run the following command.

```
sudo apt update; sudo apt upgrade
```

Step 2: Install PostgreSQL Database Server and the PostGIS Extension

We will use PostgreSQL to store map data. PostGIS is a geospatial extension to PostgreSQL. Run the following commands to install them.

```
sudo apt install postgresql postgresql
-contrib postgis postgresql-10-postgis
-2.4
```

PostgreSQL database server will automatically start and listens on `127.0.0.1:5432`. The `postgres` user will be created on the OS during the installation process. It's the super user for PostgreSQL database server. By default, this user has no password and there's no need to set one because you can use `sudo` to switch to the `postgres` user and log into PostgreSQL server.

```
sudo -u postgres -i
```

Now you can create a PostgreSQL database user `osm`.

```
createuser osm
```

Then create a database named `gis` and at the same time make `osm` as the owner of the database. `-E UTF8` specifies the character encoding scheme to be used in the database is UTF8.

```
createdb -E UTF8 -O osm gis
```

Next, create the `postgis` and `hstore` extension for the `gis` database.

```
psql -c "CREATE EXTENSION postgis;" -d
gis

psql -c "CREATE EXTENSION hstore;" -d
gis
```

Set `osm` as the table owner.

```
psql -c "ALTER TABLE spatial_ref_sys O
WNER TO osm;" -d gis
```

Exit from the `postgres` user.

```
exit
```

Create `osm` user on your operating system so the tile server can run as `osm` user. The following command will create a system user without password.

```
sudo adduser --system osm
```

Step 3: Download Map Stylesheet and Map Data

Change to osm’s home directory.

```
cd /home/osm/
```

Download the latest CartoCSS map stylesheets to the `osm` user’s home directory with git.

```
sudo apt install git
```

```
git clone https://github.com/gravityst
orm/openstreetmap-carto.git
```

If you see “permission denied” error while running the above command, then you can grant permissions with the following command. Replace `username` with your real username.

```
sudo setfacl -R -m u:username:rwx /hom
e/osm/
```

Next, run the following command to download the map data of the whole planet (50G) in PBF (ProtoBufBinary) format.

```
wget -c http://planet.openstreetmap.or
g/pbf/planet-latest.osm.pbf
```

Note that download speeds for openstreetmap.org are currently restricted to 2048 KB/s. You can download the plant map from another mirror, like

```
wget -c https://download.bbbike.org/os
m/planet/planet-latest.osm.pbf
```

If you want a map of individual country/state/province/city, go to http://download.geofabrik.de. Also, BBBike.org provides extracts of more than 200 cities and regions worldwide in different formats. For example, download the map data of Great Britain (1.1G) with the following command.

```
wget -c http://download.geofabrik.de/e
urope/great-britain-latest.osm.pbf
```

Step 4: Optimize PostgreSQL Server Performance

The import process can take some time. To speed up this process, we can tune some PostgreSQL server settings to improve performance. Edit PostgreSQL main configuration file.

```
sudo nano /etc/postgresql/10/main/postgresql.conf
```

First, we should change the value of `shared_buffers`. The default setting is:

```
shared_buffers = 128MB
```

This is too small. The rule of thumb is to set it to 25% of your total RAM (excluding swap space). For example, my VPS has 60G RAM, so I set it to:

```
shared_buffers = 15GB
```

Find the following line.

```
#work_mem = 4MB
#maintenance_work_mem = 64MB
```

Again, the value is too small.

```
work_mem = 1GB
maintenance_work_mem = 8GB
```

Then find the following line.

```
#effective_cache_size = 4GB
```

If you have lots of RAM like I do, you can set a higher value for the effective_cache_size like 20G.

```
effective_cache_size = 20GB
```

Save and close the file. Restart PostgreSQL for the changes to take effect.

```
sudo systemctl restart postgresql
```

By default, PostgreSQL would try use huge pages in RAM. However, Linux by default does not allocate huge pages. Check the process ID of PostgreSQL.

```
sudo head -1 /var/lib/postgresql/10/main/postmaster.pid
```

Sample output:

```
7031
```

Then check the VmPeak value of this process ID.

```
grep ^VmPeak /proc/7031/status
```

Sample output:

```
VmPeak: 16282784 kB
```

This is the peak memory size that will be used by PostgreSQL. Now check the size of huge page in Linux.

```
cat /proc/meminfo | grep -i huge
```

Sample output:

```
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

We can calculate how many huge pages we need. Divide the VmPeak value by the size of huge page: 16282784 kB / 2048 kB = 7950. Edit /etc/sysctl.conf file.

```
sudo nano /etc/sysctl.conf
```

Add the following line to allocate 7950 huge pages.

```
vm.nr_hugepages = 7950
```

Save and close the file. Then apply the changes.

```
sudo sysctl -p
```

If you check the meminfo again,

```
cat /proc/meminfo | grep -i huge
```

We can see there are 7950 huge pages available.

```
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:    7950
HugePages_Free:     7950
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Restart PostgreSQL to use huge pages.

```
sudo systemctl restart postgresql
```

It’s recommended to configure SSH keepalive so that you don’t lose the SSH connection. It’s very easy to do. Just open the SSH client configuration file on your local Linux machine.

```
sudo nano /etc/ssh/ssh_config
```

And paste the following text at the end of the file.

```
ServerAliveInterval 60
```

Then save the file and connect to your Ubuntu server. You can also access the remote server via VNC to prevent flaky connection interrupting the import process.

Step 5: Import the Map Data to PostgreSQL

To import map data, we need to install `osm2pgsql` which converts OpenStreetMap data to postGIS-enabled PostgreSQL databases.

```
sudo apt install osm2pgsql
```

Grant permissions to the postgres user.

```
sudo setfacl -R -m u:postgres:rwx /home/osm/
```

Switch to the `postgres` user.

```
sudo -u postgres -i
```

Run the following command to load map stylesheet and map data into the `gis` database. Replace `great-britain-latest.osm.pbf` with your own map data file.

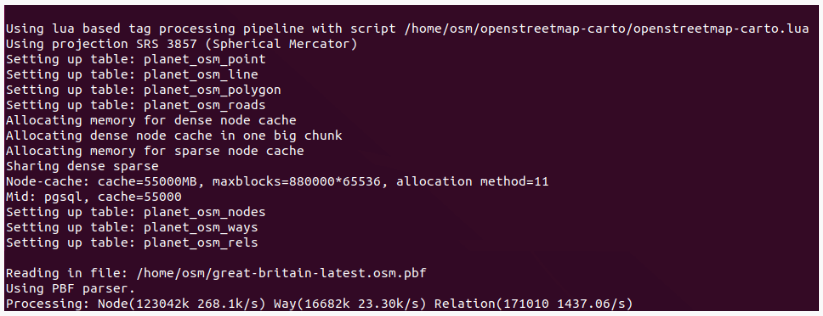
```
osm2pgsql --slim -d gis --hstore --multi-geometry --number-processes 10 --tag-transform-script /home/osm/openstreetmap-carto/openstreetmap-carto.lua --style /home/osm/openstreetmap-carto/openstreetmap-carto.style -C 32000 /home/osm/great-britain-latest.osm.pbf
```

where

- `--slim`: run in slim mode rather than normal mode. This option is needed if you want to update the map data using OSM change files (OSC) in the future.
- `-d gis`: select database.
- `--hstore`: add tags without column to an additional hstore (key/value) column to PostgreSQL tables
- `--multi-geometry`: generate multi-geometry features in postgresql tables.
- `--style`: specify the location of style file
- `--number-processes`: number of CPU cores on your server. I have 10.

- `-C` flag specifies the cache size in MegaBytes. It should be around 70% of the free RAM on your machine. Bigger cache size results in faster import speed. For example, my server has 60GB RAM, so I can specify `-C 32000`. Be aware that PostgreSQL will need RAM for shared_buffers. Use this formula to calculate how big the cache size should be: `(Total RAM - PostgreSQL shared_buffers) * 70%`
- Finally, you need to specify the location of map data file.

Command Output:



```
Using lua based tag processing pipeline with script /home/osm/openstreetmap-carto/openstreetmap-carto.lua
Using projection SRS 3857 (Spherical Mercator)
Setting up table: planet_osm_point
Setting up table: planet_osm_line
Setting up table: planet_osm_polygon
Setting up table: planet_osm_roads
Allocating memory for dense node cache
Allocating dense node cache in one big chunk
Allocating memory for sparse node cache
Sharing dense sparse
Node-cache: cache=55000MB, maxblocks=880000*65536, allocation method=11
Mid: pgsql, cache=55000
Setting up table: planet_osm_nodes
Setting up table: planet_osm_ways
Setting up table: planet_osm_rels

Reading in file: /home/osm/great-britain-latest.osm.pbf
Using PBF parser.
Processing: Node(123042k 268.1k/s) Way(16682k 23.30k/s) Relation(171010 1437.06/s)
```

If you are going to import the full planet map data, then use the `--drop` option and the `--flat-nodes` option to increase the import speed. Note that the `--flat-nodes` option isn't suitable for small maps.

```
osm2pgsql --slim -d gis --drop --flat-nodes nodes.cache --hstore --multi-geometry --number-processes 10 --tag-transform-script /home/osm/openstreetmap-carto/openstreetmap-carto.lua --style /home/osm/openstreetmap-carto/openstreetmap-carto.style -C 32000 /home/osm/planet-latest.osm.pbf
```

RAM usage will gradually increase during the importing process. Once the import is complete, grant all privileges of the `gis` database to the `osm` user.

```
psql -c "GRANT ALL PRIVILEGES ON ALL T
ABLES IN SCHEMA public TO osm;" -d gis
```

Exit from the `postgres` user.

```
exit
```

Step 6: Install mod_tile and Renderd

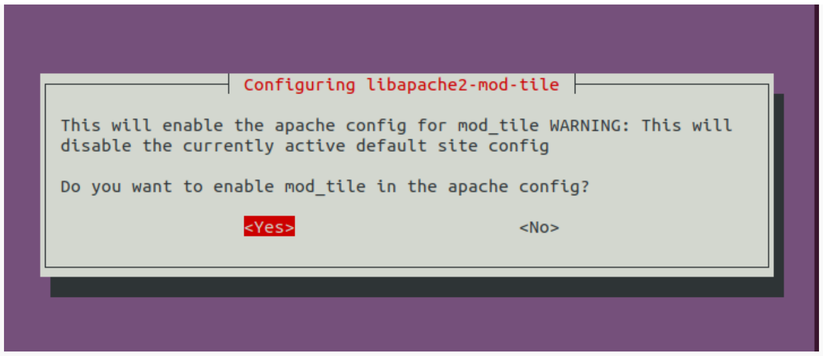
`mod_tile` is an Apache module that is required to serve tiles and `renderd` is the rendering daemon for rendering OpenStreetMap tiles. The default Ubuntu repository does not include `mod_tile` and `renderd`, but we can install them from the OSM PPA.

```
sudo apt install software-properties-c
ommon

sudo add-apt-repository ppa:osmadmins/
ppa

sudo apt install libapache2-mod-tile r
enderd
```

During the installation, it will install Apache web server and ask if you want to enable mod_tile in the Apache config. Select Yes and press Enter. This will create an Apache config file for mod_tile (`/etc/apache2/sites-available/tileserver_site.conf`).



```
Configuring libapache2-mod-tile

This will enable the apache config for mod_tile WARNING: This will
disable the currently active default site config

Do you want to enable mod_tile in the apache config?

<Yes>                                   <No>
```

The render daemon will automatically start, as can be seen with:

```
systemctl status renderd
```

Step 7: Generate Mapnik Stylesheet

Install the required packages.

```
sudo apt install curl unzip gdal-bin m
apnik-utils libmapnik-dev python3-pip
```

We also need to install nodejs and npm from the upstream repository with the following commands.

```
curl -sL https://deb.nodesource.com/se
tup_12.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Then install the carto package with npm.

```
sudo npm install -g carto
```

Install the `psycopg2` Python module.

```
sudo -H pip3 install psycopg2
```

Switch to the `postgres` user.

```
sudo -u postgres -i
```

Cd into the carto style directory.

```
cd /home/osm/openstreetmap-carto/
```

Get shapefiles.

```
scripts/get-external-data.py
```

If you encounter the following error message while running the above command, then you have DNS issues. Simply wait for several minutes and run the Python script again.

```
Failed to establish a new connection:
[Errno -3] Temporary failure in name r
esolution
```

Now build the Mapnik XML stylesheet with the `carto` map stylesheet compiler.

```
carto project.mml > style.xml
```

Grant all privileges of the `gis` database to the `osm` user.

```
psql -c "GRANT ALL PRIVILEGES ON ALL T
ABLES IN SCHEMA public TO osm;" -d gis
```

Exit from the `postgres` user.

```
exit
```

Step 8: Install Fonts

You need to install the `ttf-dejavu` package.

```
sudo apt install ttf-dejavu
```

To display non-Latin characters, install the following packages.

```
sudo apt install fonts-noto-cjk fonts-
noto-hinted fonts-noto-unhinted ttf-un
ifont
```

Step 9: Configure renderd

Edit renderd config file.

```
sudo nano /etc/renderd.conf
```

In the `[renderd]` section, change the number of threads according to the number of CPU cores on your server.

```
num_threads=10
```

In the `[default]` section, change the value of XML and HOST to the following. Note that lines beginning with semicolons (;) are comments.

```
XML=/home/osm/openstreetmap-carto/styl
e.xml
HOST=map.your-domain.com
```

In `[mapnik]` section, change the value of `plugins_dir` to the following.

```
plugins_dir=/usr/lib/mapnik/3.0/input/
```

You can print the default input plugins directory with the following command.

```
mapnik-config --input-plugins
```

If you want to display non-Latin characters, it's better to change the font settings to the following.

```
font_dir=/usr/share/fonts/truetype
font_dir_recurse=true
```

Save and close the file. Then edit the init script file

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

Find the following line.

```
RUNASUSER=www-data
```

Change the user to `osm`. This is needed to load map data from PostgreSQL database.

```
RUNASUSER=osm
```

Save the file. Set `osm` as the owner of `/var/lib/mod_tile/` directory, which will hold the rendered tile files.

```
sudo chown osm /var/lib/mod_tile/ -R
```

Then restart renderd service.

```
sudo systemctl daemon-reload

sudo systemctl restart renderd
```

You need to check the log of renderd.

```
sudo journalctl -eu renderd
```

Make sure renderd does not produce any error in the log, or the map won't be displayed.

Step 10: Configure Apache

Edit the OSM virtual host file.

```
sudo nano /etc/apache2/sites-availabl
e/tileserver_site.conf
```

Change the ServerName to your own domain name like `map.yourdomain.com`. You also need to create DNS A record for this sub-domain.

```
ServerName map.yourdomain.com
```

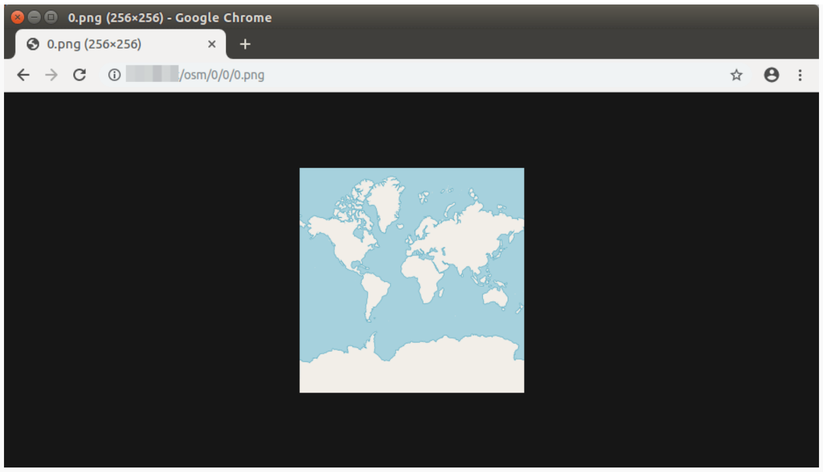
Save and close the file. Restart Apache.

```
sudo systemctl restart apache2
```

Then in your web browser address bar, type

```
map.your-domain.com/osm/0/0/0.png
```

You should see the tile of the world map. Congrats! You just successfully built your own OSM tile server.



If you have enabled the UFW firewall, be sure to open port 80 and 443 with the following command.

```
sudo ufw allow 80,443/tcp
```

Step 11: Display Your Tiled Web Map

Tiled web map is also known as **slippy map** in OpenStreetMap terminology. There are two free and open-source JavaScript map libraries you can use for your tile server: **OpenLayer** and **Leaflet**. The advantage of Leaflet is that it is simple to use and your map will be mobile-friendly.

OpenLayer

To display your slippy map with OpenLayer, download JavaScript and CSS from openlayer.org and extract it to the webroot folder.

```
cd /var/www/

sudo wget https://github.com/openlayers/openlayers/releases/download/v5.3.0/v5.3.0.zip

sudo unzip v5.3.0.zip
```

Next, create the `index.html` file.

```
sudo nano /var/www/index.html
```

Paste the following HTML code in the file. Replace red-colored text and adjust the longitude, latitude and zoom level according to your needs.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Accessible Map</title>
<link rel="stylesheet" href="http://map.yourdomain.com/v5.3.0/css/ol.css" type="text/css">
<script src="http://map.yourdomain.com/v5.3.0/build/ol.js"></script>
<style>
  a.skiplink {
    position: absolute;
    clip: rect(1px, 1px, 1px, 1px);
    padding: 0;
    border: 0;
    height: 1px;
    width: 1px;
    overflow: hidden;
  }
  a.skiplink:focus {
    clip: auto;
    height: auto;
    width: auto;
    background-color: #fff;
    padding: 0.3em;
  }
  #map:focus {
    outline: #4A74A8 solid 0.15em;
  }
</style>
</head>
<body>
  <a class="skiplink" href="#map">Go t
o map</a>
  <div id="map" class="map" tabindex
="0"></div>
  <button id="zoom-out">Zoom out</butt
on>
  <button id="zoom-in">Zoom in</button
>
  <script>
    var map = new ol.Map({
      layers: [
        new ol.layer.Tile({
          source: new ol.source.OSM({
             url: 'http://map.yourdoma
in.com/osm/{z}/{x}/{y}.png'
          })
        })
      ],
      target: 'map',
      controls: ol.control.defaults({
        attributionOptions: /** @type
 {olx.control.AttributionOptions} */
 ({
          collapsible: false
        })
      }),
    view: new ol.View({
       center: [244780.24508882355, 73
86452.183179816],
       zoom:5
    })
 });

  document.getElementById('zoom-out').
onclick = function() {
    var view = map.getView();
    var zoom = view.getZoom();
    view.setZoom(zoom - 1);
  };

  document.getElementById('zoom-in').o
nclick = function() {
      var view = map.getView();
      var zoom = view.getZoom();
      view.setZoom(zoom + 1);
  };
</script>
</body>
</html>
```

Save and close the file. Now you can view your slippy map by typing your sub-domain in the browser address bar.

```
map.yourdomain.com
```

or

```
map.yourdomain.com/index.html
```

Leaflet

To display your slippy map with Leftlet, download JavaScript and CSS from leftletjs.com and extract it to the webroot folder.

```
cd /var/www/

sudo wget http://cdn.leafletjs.com/lea
flet/v1.6.0/leaflet.zip

sudo unzip leaflet.zip
```

Next, create the `index.html` file.

```
sudo nano /var/www/index.html
```

Paste the following HTML code in the file. Replace red-colored text and adjust the longitude, latitude and zoom level according to your needs.

```
<html>
<head>
<meta charset="UTF-8">
<title>My first osm</title>
<link rel="stylesheet" type="text/css"
href="leaflet.css"/>
<script type="text/javascript" src="le
aflet.js"></script>
<style>
   #map{width:100%;height:100%}
</style>
</head>

<body>
  <div id="map"></div>
  <script>
    var map = L.map('map').setView([5
3.555,9.899],5);
    L.tileLayer('http://map.yourdomai
n.com/osm/{z}/{x}/{y}.png',{maxZoom:1
8}).addTo(map);
</script>
</body>
</html>
```

Save and close the file. Now you can view your slippy map by typing your server IP address in browser.

```
map.yourdomain.com
```

or

```
map.yourdomain.com/index.html
```

Step 12: Pre-render Tiles

Rendering tiles on-the-fly will increase the map loading time in web browser. To pre-render tiles instead of rendering on the fly, use the following `render_list` command. Use `-z` and `-`

`Z` flag specify the zoom level and replace the number of threads according to the number of CPU cores on your server. `Render_list` renders a list of map tiles by sending requests to the rendering daemon. Pre-rendered tiles will be cached in `/var/lib/mod_tile` directory.

```
render_list -m default -a -z 0 -Z 19 -
-num-threads=10
```

If later you updated the map data, you can pre-render all tiles again by using the `--force` option.

```
render_list -m default -a -z 0 -Z 19 -
-num-threads=10 --force
```

To render map tiles in the background, add the `&` symbol at the end.

```
render_list -m default -a -z 0 -Z 19 -
-num-threads=10 &
```

Now you can close the terminal window. To check the rendering progress, open another SSH session, and run the following command.

```
sudo journalctl -eu renderd
```

The above command will show the latest log of the `renderd` service. The following lines show that my OSM server is now rendering map tiles at zoom level 12.

```
 renderd[20838]: DEBUG: START TILE def
ault 12 1008-1015 4056-4063, new metat
ile
```

```
 renderd[20838]: Rendering projected c
oordinates 12 1008 4056 -> -10175297.2
05328|-19724422.274944 -10097025.68836
4|-19646150.757980 to a 8 x 8 tile
 renderd[20838]: DEBUG: DONE TILE defa
ult 12 1008-1015 3984-3991 in 0.799 se
conds
 renderd[20838]: DEBUG: Sending render
cmd(3 default 12/1008/3984) with proto
col version 2 to fd 18
 renderd[20838]: DEBUG: Got incoming r
equest with protocol version 2
 renderd[20838]: DEBUG: Got command Re
nderBulk fd(18) xml(default), z(12), x
(1008), y(4064), mime(image/png), opti
ons()
 renderd[20838]: DEBUG: START TILE def
ault 12 1008-1015 4064-4071, new metat
ile
 renderd[20838]: Rendering projected c
oordinates 12 1008 4064 -> -10175297.2
05328|-19802693.791908 -10097025.68836
4|-19724422.274944 to a 8 x 8 tile
```

Step 13: Enable HTTPS

To encrypt HTTP traffic, we can obtain and install a free TLS certificate from Let's Encrypt. First, install the Let's Encrypt client (certbot) on Ubuntu 18.04.

```
sudo add-apt-repository ppa:certbot/ce
rtbot

sudo apt install certbot
```

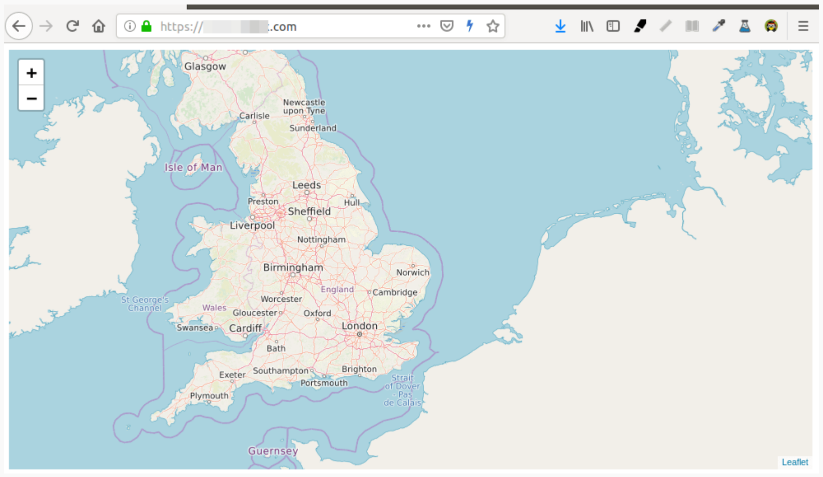
Since we are using Apache web server, we also need to install the Apache plugin.

```
sudo apt install python3-certbot-apach
e
```

Then run the following command to obtain and install TLS certificate.

```
sudo certbot --apache --agree-tos --re
direct --hsts --staple-ocsp --must-sta
ple --email your-account@example.com -
d map.yourdomain.com
```

Once the certificate is installed, refresh the web page and you will see a lock in the address bar.



If you see a yellow triangle in Firefox address bar, that means the tile URLs are still using HTTP. You need to edit the index.html file and replace all HTTP protocol with HTTPS with the following command.

```
sudo sed -i 's/http/https/g' /var/www/
index.html
```

Step 14: Enable HTTP2

To further improve map loading performance, you can enable HTTP2 protocol. First, you need to enable the HTTP2 module.

```
sudo a2enmod http2
```

Then open the SSL virtual host file.

```
sudo nano /etc/apache2/sites-enabled/t
ileserver_site-le-ssl.conf
```

Put the following directive after the opening `<VirtualHost *:443>` tag.

```
Protocols h2 http/1.1
```

Save and close the file. Then restart Apache for the changes to take effect.

```
sudo systemctl restart apache2
```

Restrict Access to Your OSM Tile Server

By default, anyone can use OpenLayer or Leaflet to create a slippy map with the URL of your tile server. To restrict access to your tile server, edit the Apache virtual host file.

```
sudo nano /etc/apache2/sites-enabled/t
ileserver_site-le-ssl.conf
```

Add the following lines in the `<VirtualHost>` tags.

```
    <Location /osm>
        SetEnvIf Referer example\.com
 trusted_referer
        Order deny,allow
        Deny from all
        Allow from env=trusted_referer
    </Location>
```

The above code checks if the HTTP referer header includes your own domain. If not, access to the `/osm` directory will be denied. The backslash is used to escape the dot character. To add multiple hostnames as trusted refererrs, use the following syntax.

```
SetEnvIf Referer (example\.com|www\.ex
ample\.com|map\.example\.com) trusted_
referer
```

Save and close the file. Then test the syntax.

```
sudo apache2ctl -t
```

If the syntax is Ok, reload Apache for the changes to take effect.

```
sudo systemctl reload apache2
```

Auto-Renew TLS Certificate

You can create Cron job to automatically renew TLS certificate. Simply open root user’s crontab file.

```
sudo crontab -e
```

Add the following line at the bottom of the file.

```
@daily certbot renew --quiet && system
ctl reload apache2
```

PostgreSQL Database and Web Server on Different Hosts

If your PostgreSQL and Apache web server reside on different hosts, then you need to edit the `project.mml` file on the Apache host.

```
nano /home/osm/openstreetmap-carto-4.2
0.0/project.mml
```

Find the following lines:

```
osm2pgsql: &osm2pgsql
  type: "postgis"
  dbname: "gis"
  key_field: ""
  geometry_field: "way"
  extent: "-20037508,-20037508,2003750
8,20037508"
```

Specify the IP address of PostgreSQL database server.

```
osm2pgsql: &osm2pgsql
  type: "postgis"
  host: "10.0.0.2"
  dbname: "gis"
  key_field: ""
  geometry_field: "way"
  extent: "-20037508,-20037508,2003750
8,20037508"
```

Save and close the file. Then build the Mapnik XML stylesheet with the `carto` map stylesheet compiler.

```
carto project.mml > style.xml
```

On the PostgreSQL database server, edit the main configuration file.

```
sudo nano /etc/postgresql/10/main/post
gresql.conf
```

Add the following line to set PostgreSQL to listen on all interfaces.

```
listen_addresses = '*'
```

Save and close the file. Then edit the PostgreSQL client authentication configuration file.

```
sudo nano /etc/postgresql/10/main/pg_h
ba.conf
```

Add the following line at the end of the file to allow the `osm` user to login from the Apache host. Replace 10.0.0.1 with the IP address of Apache host.

```
host   gis   osm   10.0.0.1/32   trust
```

Save and close the file. Then restart PostgreSQL.

```
sudo systemctl restart postgresql
```

Restart the render daemon on the Apache host.

```
sudo systemctl restart renderd
```

You need to check the log of renderd. Make sure renderd does not produce any error in the log, or the map won't be displayed.

```
sudo journalctl -eu renderd
```

You should also restrict access to port 5432 of the PostgreSQL database server. For example, you can use the following UFW command to allow the IP address of Apache host only.

```
sudo ufw allow in from 10.0.0.1 to any port 5432
```

Conclusion

I hope this tutorial helped you set up OpenStreetMap tile server on Ubuntu 18.04. As always, if you found this post useful, then subscribe to our free newsletter to get more tips and tricks. Take care 🙂

Rate this tutorial

★★★★★ [Total: 10 Average: 4.7]

Linux OpenStreetMap Self Hosted Ubuntu

Ubuntu Server