Creating offline map server on local machine (Windows 8/10):

Software tools required:

- Mapnik SDK version 2.2 (For windows)
- postgres database with postgis (Version 9.4 at this time)
- Osm2pgsql
- OSM map data
- Tilecache
- Leaflet API

Step 1:

First download osm data from geofabrik. In my case Nepal-osm.pbf from http://download.geofabrik.de/asia/nepal.html.

Step2:

Download and install mapnik for windows.

http://mapnik.org/pages/downloads.html . download windows sdk mapnik 2.2. package. Version 3 will not work. Installation guide: https://github.com/mapnik/mapnik/wiki/WindowsInstallation

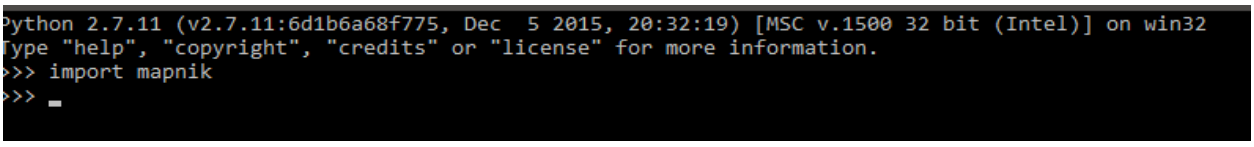
Unzip mapnik and set the lib and bin folder to environment variable.

For PYTHON support add:

- PYTHON 2.7: C:\mapnik-v2.2.0\python\2.7\site-packages; to the PYTHONPATH variable.

Now, you can check mapnik by sampling running python and importing mapnik. If no error, you have successfully installed mapnik.

Run python: import Mapnik



There is simple guide to create your first mapnik in above installation guide.

Step 3:

Install postgres database and install postgis and create a new database name ‘osm’.

Download osm2pgsql. Note that osm2pgsql only work with Cygwin so download Cygwin version and run it. Now, Import the download nepal-latest.osm.pbf file in the database by following steps.

Run the osm2pgsql.exe file. If there is error in running this exe file create a shortcut and run.

Command:

```
osm2pgsql -H "localhost" -P 5432 -W -U postgres -S "C:/Osm2pgsql-cygwin-package/default.style"
-r pbf --bbox 85.2755,27.6543,85.361,27.7452 -d osm -j "C:/nepal-latest.osm.pbf"
```

Where:

- H – host address of your machine
- P – Port for Postgres database
- W - Force password prompt
- U – User for postgres database
- S - for Stylesheet for database (can find default on OSM2Pgsql folder)
- r – specify format of map data (pbf)
- bbox – Area boundary to extract map data
- d – Database name
- j – Specify path to the map file.

Step 4:

Now we have data in our database, so next is to use that data and convert it into map using mapnik. First we need some additional data to be downloaded. The osm styles. Carto or original osm style sheets. In addition, you have to download shape files as mention in these styles. The finally make style.xml.

Carto:

Installation in python pip: pip install carto

Run: carto project.mml > osm.xml //osm.xml is our filename and can be any.

Here project.mml is the template from which osm.xml will be generated. Note that there is no way to specify username dbname password etc. The only thing we can do is add new section for password, host and user in the mapnik.xml file. Here is simple hack:

Search ctrl+F <Parameter name="dbname"><![CDATA[“”]]></Parameter> which will be blank. Now replace all <Parameter name="dbname"><![CDATA[“”]]></Parameter> with

<Parameter name="host"><![CDATA[localhost]]></Parameter>

<Parameter name="port"><![CDATA[5432]]></Parameter>

<Parameter name="username"><![CDATA[postgres]]></Parameter>

<Parameter name="password"><![CDATA[Apple123]]></Parameter>

<Parameter name="dbname"><![CDATA[osm]]></Parameter>

Mapnik stylesheet (Used this method for this project):

Installation download from github: https://github.com/openstreetmap/mapnik-stylesheets

Now, download the shape files and create a new world boundaries folder. And inside this folder put all shape files (not folder so that all file will be in one single folder).

Then create osm.xml file: generate_xml.py osm.xml ktm.xml --dbname osm --host localhost --user postgres --port 5432 --password pg123 --estimate_extent true --epsg=900913

Note that this stylesheet default projection is 4326 and osm is 3857 so we need to use 900913 as projection epsg. Note that 900913 is subset of EPSG:3857.

EPSG:3857 – Map projection like a flat surface. Used by googlemap and osm.

EPSG:4326 – Map projection like a sphere. Used by googleearth.

In addition, estimate extent means automatically estimate the bbox of map that is present in the database.

Step 5:

Install Tilecache:

Download it via pip install. Pip install tilecache or download it directly and unzip it. Direct download preferable as it contain .py files for creating standalone server "tilecache_http_server.py" and seeding "tilecache_seed.py" etc. Here, installation means only unzipping file so the setup.py mean nothing.

Also, we need a standalone http server to access localhost:8080\1.0.0\osm\x\y\z.png. So install **Paste** from pip. Reference in documentation. http://tilecache.org/docs/README.html

Now edit the tilecache.cfg file:

Add the following cache section base which is cache location directory.

```
[cache]
type=Disk
# must not be base=/C:/Users/Ranjit Kaliraj/tilecache/Cache otherwise error will thrown
base= C:\Users\Ranjit Kaliraj\tilecache\Cache
```

Here, osm is just a name of layer and there is no relation to style or database. So, add the style that is generated from mapnik stylesheet (reference not copy it to this folder otherwise the xml file will not able to access those shapefiles, just simply provide path address to that xml file). Here, spherical Mercator and srs EPSG 3857 is same. Level 20 means zoom level. Enabling metatile will throw error such as image not found, so disable it.

```
[osm]
type=Mapnik
#levels mean zoom level
levels=20
mapfile= C:\ stylesheets\ktm.xml
spherical_mercator=true
# srs=EPSG:3857
# metatile=yes
maxResolution=234.375
# bbox=85.2270,27.5932,85.4233,27.7717
# extent_type=loose

Uncomment configurations in order to enable it such as bbox.
```

maxResolution examples:

Using custom projections with TileCache, Mapnik and Leaflet

Posted by Bjørn Sandvik

After five blog posts (1, 2, 3, 4, 5) we finally have a terrain map of Jotunheimen, a mountainous area in Norway with beautiful lakes and glaciers. It's time to publish the map on the web. In this blog post I'll show how you can use TileCache and Mapnik to render map tiles, and how to load these tiles into Leaflet using the UTM coordinate system.

To make a slippy map - a zoomable and draggable map - we need to serve map tiles instead of the large map image we created in the last blog post. The original Digital Elevation Model (DEM) is 3134 x 3134 pixels, and with a bit of upscaling we can using this tiling scheme:

Zoom	Map size	Tiles	Resolution
0	256 x 256 px	1 x 1 = 1	234.375
1	512 x 512 px	2 x 2 = 4	117.1875
2	1024 x 1024 px	4 x 4 = 16	58.59375
3	2048 x 2048 px	8 x 8 = 64	29.296875
4	4096 x 4096 px	16 x 16 = 256	14.6484375

Each map tile is 256 x 256 pixels. We need one map tile to cover the first zoom level and 256 tiles to cover the last zoom level. The original map area is 60.000 x 60.000 meters, and the resolution shows meters per pixel (60.000 / 256 = 234.375). We have enough information to configure TileCache (tilecache.cfg):

```
[jotunheimen]
type=Mapnik
src=EPSG:32632
bbox=432000,6790000,492000,6850000
maxResolution=234.375
```

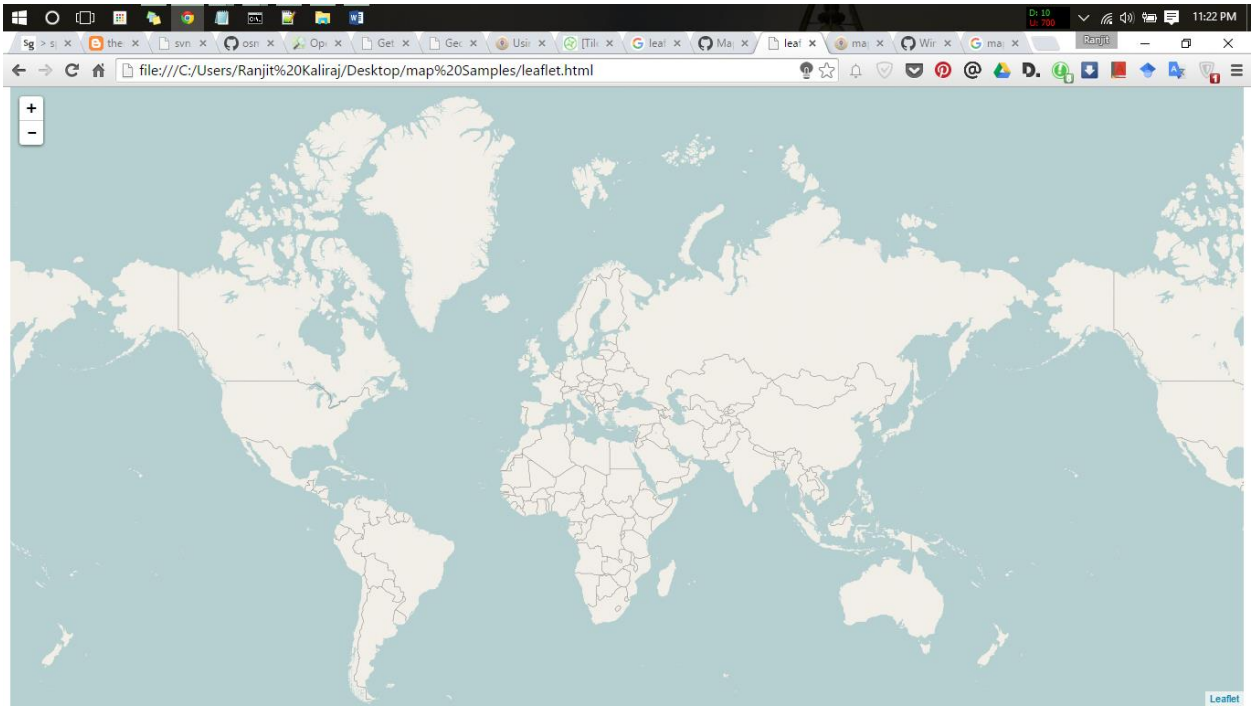
Now the tilecache configuration file is successfully setup. Now, run the tilecache_http_server.py file and test in browser: localhost:8080/1.0.0/osm.0/0/0/png and you will see the world map picture. Here, osm is name of layer mention in the config file. 0/0/0.png (first 0 is longitude, and latitude and finally zoom level). Now, we have displayed static map now its tile to make it interactive in following step below.

Step 6:

We need to use slippy map in order to enable interaction. Here, I have used leaflet api. It is a javascript api. First download, api and create a simple html file with following code.

```
<html>
<head>
	<link rel='stylesheet' href='leaflet/leaflet.css' />
	<script src='leaflet/leaflet.js' type='text/javascript'></script>
	<style type="text/css">
		html, body {height: 100%;overflow: hidden;}
		#map {height: 100%;}
	</style>
</head>
<body>
	<div id="map"></div>
	<script>

	var map = new L.Map('map', {
		center: [0, 0],
		zoom: 0
	});

	//var marker = L.marker([85.2955,27.6943]).addTo(map);
	//marker.bindPopup("<b>Hello world!</b><br>I am a popup.").openPopup();

	L.tileLayer('http://localhost:8080/1.0.0/osm/{z}/{x}/{y}.png',{tms: true}).addTo(map);

	//map.setView([0, 0], 0, {reset: true});

	//the following avoid not displaying map when start untill zoom in or out
	map.setView([52.1, 4], 3, {animate: false});

	</script>
</body>
</html>
```

Now run this html in browser and yes, you will be able to view map.



Some Leaflet problem:

Tile is unaligned as below:

Solved by using tms: true

Available at:

http://gis.stackexchange.com/questions/48121/add-tilecache-schema-layer-to-leaflet-map?rq=1

L.tileLayer('http://localhost:8080/1.0.0/osm/{z}/{x}/{y}.png',{tms: true}).addTo(map);

Another problem: Map is not displaying when first opening browser until zoom in or out.

map.setView([52.1, 4], 3, {animate: false});

Available at: https://github.com/Leaflet/Leaflet/issues/3002