**MAPSERVER**

**Documentation**

*By Infra team*

**Contents**

**Name Pagenumber**

**Introduction** 03

**Mapserver** 04 - 14

**Documentation of Geoserver-Spatial data RUN as Docker** 15 – 17

Openstreetmap-tile-server up and running 15

Setup Postgresql DB and GUI 15

Deploy Geoserver and link the postgres db with geoserver17

**RUN.sh** 18 - 19

**Introduction**

Introduce a self-esteemed indigenous MAPSERVER for Kerala Police and its work, which is useful for Kerala Police in various means. The infra team of CCTNS implement an indigenous Map server with a demo version. The demo server works with our concerns and a complete Documentation of the server is described on the next page with every bit of change on the Dockerfile, run.sh files also. The script will also include for ready reference and easy troubleshoot.

The main concerns of this document is to identify the content and its usage. And the script added here for easy understanding of the codes used inside the Dockerfile, run.sh files.

**MAPSER­VER**

1. Create a directory called ‘Mapserver’ as the project directory.

$ cd Mapserver, This will be the base project directory

Download/ Clone the openstreetmap tile server github project from github.com. The link of the github repository is mentioned here for reference https://github.com/Overv/openstreetmap-tile-server.git . This is the base file and other addons are in the main file for customising indegeneous docker image for Police Department.

$ mv openstreetmap-tile-server Openstreetmap

2. Import the data files

needed a data file of a country to generate the tiles. Here, I have used pbf data files for that(osm.pbf extension). Download those pbf files from http://download.geofabrik.de website. I have downloaded and imported the pbf file of Malta because its’s size is relatively small. It may take some time to load the large pbf files. Additionally to the pbf files, import the poly file of the country. Create a directory called ‘Data’ under the ‘Openstreetmap’ directory and place data files there. create an empty file called ‘data.poly’ under the Data directory.

$ cd ~/Mapserver/Openstreetmap/

$ mkdir Data

$ cd Data

$ wget http://download.geofabrik.de/europe/malta-latest.osm.pbf

$ touch data.poly

$ nano data.poly

3. Add external CSS and javascript files

To load the map offline in local machhine, add CSS and javascript files externally. Here, leaflet js is used to display the map. Download leaflet js version 1.6.0 and place the files leaflet.css and leaflet.js to the directory path ‘~/Mapserver/Openstreetmap/’. make changes in the leaflet-demo.html file.

leaflet-demo.html

=================

<!DOCTYPE html>

<html>

<head>

<title>Custom Tile Server</title>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="leaflet.css" crossorigin=""/>

<script src="leaflet.js" crossorigin=""></script>

<style>

html, body, #map {

width: 100%;

height: 100%;

margin: 0;

padding: 0;

}

</style>

</head>

<body>

<div id="map"></div>

<script>

var map = L.map('map').setView([0, 0], 3);

L.tileLayer('/tile/{z}/{x}/{y}.png', {

maxZoom: 18,

attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>',

id: 'base'

}).addTo(map);

</script>

</body>

</html>

4.Edit the Dockerfile and shell script files

edit the Dockerfile for customisation. Under the ‘Install dependencies’ section of the current Dockerfile, you may see a list of dependencies that need to install through docker. I have put those dependencies into a separate file called ‘dependencies.sh’ and imported it to the Dockerfile as following.

$ cd ~/Mapserver/Openstreetmap/

$ touch dependencies.sh

$ nano dependencies.sh

dependencies.sh

===============

#!/bin/bash

apt-get update \

&& apt-get install sudo wget gnupg2 lsb-core -y \

&& wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add - \

&& echo "deb [ trusted=yes ] http://apt.postgresql.org/pub/repos/apt/ `lsb\_release -cs`-pgdg main" | tee /etc/apt/sources.list.d/pgdg.list \

&& sudo apt-get update -q \

&& sudo apt-get install -y apt-transport-https ca-certificates \

&& sudo apt-get install -qy --no-install-recommends --allow-unauthenticated \

software-properties-common \

libboost-dev \

libboost-filesystem-dev \

libboost-program-options-dev \

libboost-python-dev \

libboost-regex-dev \

libboost-system-dev \

libboost-thread-dev \

subversion \

git-core \

tar \

unzip \

wget \

bzip2 \

build-essential \

autoconf \

libtool \

libxml2-dev \

libgeos-dev \

libpq-dev \

libbz2-dev \

munin-node \

munin \

libprotobuf-c-dev \

protobuf-c-compiler \

protobuf-compiler \

python-protobuf \

libc6-dev \

libgcc1 \

protobuf-c-compiler \

libfreetype6-dev \

libtiff5-dev \

libicu-dev \

libgdal-dev \

libcairo-dev \

libcairomm-1.0-dev \

apache2 \

apache2-dev \

libagg-dev \

ttf-unifont \

autoconf \

apache2 \

apache2-dev \

libtool \

libxml2-dev \

libbz2-dev \

libgeos-dev \

libgeos++-dev \

libproj-dev \

gdal-bin \

apt-utils \

mapnik-utils \

libmapnik-dev \

clang \

fonts-noto-cjk \

fonts-noto-hinted \

fonts-noto-unhinted \

gcc \

make \

libfreetype6-dev \

libicu-dev \

liblua5.3-dev \

lua5.3 \

libmapnik-dev \

autotools-dev \

automake \

npm \

nodejs \

mapnik-utils \

osmium-tool \

osmosis \

postgis \

postgresql-12 \

postgresql-server-dev-12 \

postgresql-contrib-12 \

protobuf-c-compiler \

python-is-python3 \

python3-mapnik \

python3-lxml \

python3-psycopg2 \

python3-shapely \

python3-pip \

curl \

dpkg \

sudo \

python-pycurl \

python-apt \

&& sudo apt --fix-missing install \

&& sudo apt-get clean autoclean \

&& sudo apt-get autoremove --yes \

&& ldconfig

These are the changes made in the Dockerfile also import the above dependencies.sh file with COPY and RUN commands. Also there will be small changes with the current Dockerfile because of the version may change from building of the mapserver and now.

# Based on  
# <https://switch2osm.org/manually-building-a-tile-server-18-04-lts/># Set up environment  
ENV TZ=UTC  
ENV AUTOVACUUM=off  
ENV UPDATES=disabled  
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone# Install dependenciesCOPY ./Openstreetmap/dependencies.sh /   
RUN ["chmod","+x","/dependencies.sh"]  
RUN /dependencies.sh# Set up PostGIS  
RUN wget <http://download.osgeo.org/postgis/source/postgis-3.0.0rc2.tar.gz>  
RUN tar -xvzf postgis-3.0.0rc2.tar.gz  
RUN cd postgis-3.0.0rc2 && ./configure && make && make install# Set up renderer user  
RUN adduser --disabled-password --gecos "" renderer  
USER renderer# Install latest osm2pgsql  
RUN mkdir /home/renderer/src  
WORKDIR /home/renderer/src  
RUN git clone <https://github.com/openstreetmap/osm2pgsql.git>  
WORKDIR /home/renderer/src/osm2pgsql  
RUN mkdir build  
WORKDIR /home/renderer/src/osm2pgsql/build  
RUN cmake .. \  
 && make -j $(nproc)  
USER root  
RUN make install  
RUN mkdir /nodes \  
 && chown renderer:renderer /nodes  
USER renderer# Install and test Mapnik  
RUN python -c 'import mapnik'# Install mod\_tile and renderd  
WORKDIR /home/renderer/src  
RUN git clone -b switch2osm <https://github.com/SomeoneElseOSM/mod_tile.git>  
WORKDIR /home/renderer/src/mod\_tile  
RUN ./autogen.sh \  
 && ./configure \  
 && make -j $(nproc)  
USER root  
RUN make -j $(nproc) install \  
 && make -j $(nproc) install-mod\_tile \  
 && ldconfig  
USER renderer# Configure stylesheet  
WORKDIR /home/renderer/src  
RUN git clone <https://github.com/gravitystorm/openstreetmap-carto.git> \  
 && git -C openstreetmap-carto checkout v4.23.0  
WORKDIR /home/renderer/src/openstreetmap-carto  
USER root  
RUN npm install -g carto@0.18.2  
USER renderer  
RUN carto project.mml > mapnik.xml# Load shapefiles  
USER root  
WORKDIR /home/renderer/src/openstreetmap-carto  
RUN scripts/get-shapefiles.py# Configure renderd  
USER root  
RUN sed -i 's/renderaccount/renderer/g' /usr/local/etc/renderd.conf \  
 && sed -i 's/hot/tile/g' /usr/local/etc/renderd.conf  
USER renderer# Configure Apache  
USER root  
RUN mkdir /var/lib/mod\_tile \  
 && chown renderer /var/lib/mod\_tile \  
 && mkdir /var/run/renderd \  
 && chown renderer /var/run/renderd  
RUN echo "LoadModule tile\_module /usr/lib/apache2/modules/mod\_tile.so" >> /etc/apache2/conf-available/mod\_tile.conf \  
 && echo "LoadModule headers\_module /usr/lib/apache2/modules/mod\_headers.so" >> /etc/apache2/conf-available/mod\_headers.conf \  
 && a2enconf mod\_tile && a2enconf mod\_headers  
COPY ./Openstreetmap/apache.conf /etc/apache2/sites-available/000-default.conf  
COPY ./Openstreetmap/leaflet-demo.html /var/www/html/index.html  
COPY ./Openstreetmap/leaflet.css /var/www/html/leaflet.css  
COPY ./Openstreetmap/leaflet.js /var/www/html/leaflet.js  
RUN ln -sf /dev/stdout /var/log/apache2/access.log \  
 && ln -sf /dev/stderr /var/log/apache2/error.log# Configure PosgtreSQL  
COPY ./Openstreetmap/postgresql.custom.conf.tmpl /etc/postgresql/12/main/  
RUN chown -R postgres:postgres /var/lib/postgresql \  
 && chown postgres:postgres /etc/postgresql/12/main/postgresql.custom.conf.tmpl \  
 && echo "\ninclude 'postgresql.custom.conf'" >> /etc/postgresql/12/main/postgresql.conf  
RUN echo "host all all 0.0.0.0/0 md5" >> /etc/postgresql/12/main/pg\_hba.conf \  
 && echo "host all all ::/0 md5" >> /etc/postgresql/12/main/pg\_hba.conf# copy update scripts  
COPY ./Openstreetmap/openstreetmap-tiles-update-expire /usr/bin/  
RUN chmod +x /usr/bin/openstreetmap-tiles-update-expire \  
 && mkdir /var/log/tiles \  
 && chmod a+rw /var/log/tiles \  
 && ln -s /home/renderer/src/mod\_tile/osmosis-db\_replag /usr/bin/osmosis-db\_replag \  
 && echo "\* \* \* \* \* renderer openstreetmap-tiles-update-expire\n" >> /etc/crontab# install trim\_osc.py helper script  
USER renderer  
RUN cd ~/src \  
 && git clone <https://github.com/zverik/regional> \  
 && cd regional \  
 && git checkout 612fe3e040d8bb70d2ab3b133f3b2cfc6c940520 \  
 && chmod u+x ~/src/regional/trim\_osc.py# Start running  
USER root#search for the data files in Data folder  
CMD file\_pbf=$(find Data/ -name "\*.osm.pbf" -printf "%f\n")  
CMD file\_poly=$(find Data/ -name "\*.poly" -printf "%f\n")#move the data files to the container  
RUN mkdir /home/renderer/src/Data  
COPY ./Openstreetmap/Data/$file\_pbf /home/renderer/src/Data/  
COPY ./Openstreetmap/Data/$file\_poly /var/lib/mod\_tile/file\_poly/#run script to insert data to the database  
COPY ./Openstreetmap/run.sh /  
COPY ./Openstreetmap/indexes.sql /  
ENTRYPOINT ["/run.sh"]  
CMD []#expose ports  
EXPOSE 80 5432

To import the data I use run.sh here. The data should be automatically added to the database while running if you choose an external postgres database that also possible to implement on the run.sh file. Make some changes in the current run.sh file with the following which is also include CORS (cross-origin-remote-sharing), AUTOVACCUM, cache, etc.

#!/bin/bash

set -x  
#initaialize variables

#define the number of threads  
THREADS=4

#enable CORS => CORS=1, disable CORS=> CORS=0  
ALLOW\_CORS=1

#enable or disable autovacuum feature in postgresql  
#enable autovacuum=> AUTOVACUUM=on, disable autovacuum => #AUTOVACUUM=offAUTOVACUUM=off#to enable cache assign the value  
OSM2PGSQL\_EXTRA\_ARGS="-C 4096"#create a docker volume for the databse  
sudo docker volume create openstreetmap-datafunction createPostgresConfig() {  
 cp /etc/postgresql/12/main/postgresql.custom.conf.tmpl /etc/postgresql/12/main/postgresql.custom.conf  
 sudo -u postgres echo "autovacuum = $AUTOVACUUM" >> /etc/postgresql/12/main/postgresql.custom.conf  
 #cat /etc/postgresql/12/main/postgresql.custom.conf  
}function setPostgresPassword() {  
 sudo -u postgres psql -c "ALTER USER renderer PASSWORD '${PGPASSWORD:-renderer}'"  
}# identify the data file  
 osm\_data\_file=$(find /home/renderer/src/Data/ -name "\*.osm.pbf" -printf "%f\n")# Initialize PostgreSQL  
 createPostgresConfig  
 service postgresql start  
 sudo -u postgres createuser renderer  
 sudo -u postgres createdb -E UTF8 -O renderer gis  
 sudo -u postgres psql -d gis -c "CREATE EXTENSION postgis;"  
 sudo -u postgres psql -d gis -c "CREATE EXTENSION hstore;"  
 sudo -u postgres psql -d gis -c "ALTER TABLE geometry\_columns OWNER TO renderer;"  
 sudo -u postgres psql -d gis -c "ALTER TABLE spatial\_ref\_sys OWNER TO renderer;"  
 setPostgresPassword# Import data  
 sudo -u renderer osm2pgsql -d gis --create --slim -G --hstore --tag-transform-script /home/renderer/src/openstreetmap-carto/openstreetmap-carto.lua ${OSM2PGSQL\_EXTRA\_ARGS} -S /home/renderer/src/openstreetmap-carto/openstreetmap-carto.style /home/renderer/src/Data/$osm\_data\_file# Create indexes  
 sudo -u postgres psql -d gis -f indexes.sql# Register that data has changed for mod\_tile caching purposes  
 touch /var/lib/mod\_tile/planet-import-completeservice postgresql stop# Clean /tmp  
 rm -rf /tmp/\*# Fix postgres data privileges  
 chown postgres:postgres /var/lib/postgresql -R# Configure Apache CORS  
 if [ "$ALLOW\_CORS" == "1" ]; then  
 echo "export APACHE\_ARGUMENTS='-D ALLOW\_CORS'" >> /etc/apache2/envvars  
 fi# Initialize PostgreSQL and Apache  
 createPostgresConfig  
 service postgresql start  
 service apache2 restart  
 setPostgresPassword# Configure renderd threads  
 sed -i -E "s/num\_threads=[0-9]+/num\_threads=${THREADS:-4}/g" /usr/local/etc/renderd.conf# start cron job to trigger consecutive updates  
 if [ "$UPDATES" = "enabled" ]; then  
 /etc/init.d/cron start  
 fi# Run while handling docker stop's SIGTERM  
 stop\_handler() {  
 kill -TERM "$child"  
 }  
 trap stop\_handler SIGTERMsudo -u renderer renderd -f -c /usr/local/etc/renderd.conf &  
 child=$!  
 wait "$child"service postgresql stop  
exit 0

Create a Docker-compose.yml file

cd ~/Mapserver/

touch docker-compose.yml

Add the following details to the docker-compose.yml.file

version: '3'  
services:  
osm:  
build:  
context: .  
dockerfile: ./Openstreetmap/Dockerfile  
image: "openstreetmap:latest"  
ports:  
- 80:80  
- 5432:5432  
environment:  
- HOST= 0.0.0.0

If you start the mapserver with docker-compose then, goto the Dockerfie containing folder and open the terminal and hit..

docker-compose build

docker-compose up

After the successful completion of the docker-compose file, then goto browser and check the mapserver is up and running

Check the browser with **localhost:port,**  if our mapserver works with our expectation it will show the given country coordinates and other details.

Documentation of Geoserver-Spatial data RUN as docker

Resources:

1. Openstreetmap-tile-server - overv/openstreetmap-tile-server:1.3.10

<https://github.com/Overv/openstreetmap-tile-server.git>

( We build base image of openstreetmap-tile-server from this git & make changes with the Dockerfile and build own osm tiles server and run with the newly build docker image )

1. Indian coordinate files - <https://download.geofabrik.de/malta>

(asia)

1. Postgres DB & GUI - Postgres database docker file with

postgis supported and also build a GUI

1. Stylesheet

1. <https://github.com/gravitystorm/openstreetmap-carto.git>

2. https://github.com/giggls/openstreetmap-carto-de.git

1. QGIS - deploy as docker

1. Apache Tomcat Server - as docker
2. Geoserver (MAP server)

1.

2. <https://www.mapserver.org/pdf/MapServer.pdf>

1. Leaflet

1.https://leafletjs-cdn.s3.amazonaws.com/content/leaflet/main/leaflet.zip

1. Openstreetmap-tile-server up and running
2. Build a docker image of our own with changes on the said git Dockerfiles and supporting files and note the postgres DB details on the Dockerfile and supporting file
3. import indian osm.pbf map data for the import process
4. import mapdata with the build image with a local volume

docker run --rm \

-e UPDATES=enabled \

-v /home/dev-ops/Documents/monish/Mapserver/Openstreetmap/Data/malta-latest.osm.pbf:/data.osm.pbf \

-v /home/dev-ops/Documents/monish/Mapserver/Openstreetmap/Data/malta.poly:/data.poly \

-v openstreetmap-data:/var/lib/postgresql/12/main \

-v openstreetmap-rendered-tiles:/var/lib/mod\_tile \

mstr6789/mapserver:1.0.0 \

import

1. When the import process is success(usually take 4-6 hrs depending on the hardware resource and network speed), run the image as detached mode

docker run \

-p 8989:80 \

-v openstreetmap-data:/var/lib/postgresql/12/main \

-d mstr6789/mapserver:1.0.0 \

run

1. Setup Postgresql DB and GUI

1.setup postgresql DB and Install GUI

a) docker run -p 5432:5432 \

--name postgres \

-e POSTGRES\_PASSWORD=mysecretpassword \

-e PGDATA=/var/lib/postgresql/data/pgdata \

-d cimg/postgres:13.4-postgis

NB: username and password as the tile-server docker file. i.e the way that the database connect with postgres db

b) setup GUI for the installed postgres DB

docker run -p 5050:80 \

-e "PGADMIN\_DEFAULT\_EMAIL=moni875397@gmail.com" \

-e "PGADMIN\_DEFAULT\_PASSWORD=Monish@1990" \

-d dpage/pgadmin4:latest

1. Goto browser and run call the port like this localhost:5050 and give the

credential, u r in. Then link the DB in the web gui

1. Deploy Geoserver and link the postgres db with geoserver

a) docker run -it \

--name "geoserver" \

-v /var/run/docker.sock:/var/run/docker.sock \

-v /opt/geoserver/data/:/geoserver\_data/data \

-p 8080:8080 \

-d mstr6789/geoserver:2.15.x

(NB: if the geoserver deployment was successful, goto the browser and check whether the app run well on the designed port localhost:8080. Then login the admin module for the shapefile import)

1. import shapefiles with postgis ( NB: using opensource u may need to do

this via qgis to import shape files to geoserver ) using QGIS

1. Add concerned workspace, layers, styles to the imported shape files and

calculate the coordinates. Then go to layers and preview the layer

1. docker network create qgis

docker run -d --rm --name qgis-server \

--net=qgis --hostname=qgis-server \

-v $(pwd)/data:/data:ro -p 5555:5555 \

-e "QGIS\_PROJECT\_FILE=/data/osm.qgs" \

qgis-server

**RUN.sh**

* Install git

Git might come preinstalled sometimes.

* Install Mapnik library & source

Mapnik library is used to render the Openstreetmap data into the tiles

managed by apache web server thorugh render and mod\_tile.

* Install Boost packages

Do not install *boost* from package if needed, compile *mapnik* with an

updated compiler

Notice that *boost* and *mapnik* shall be compiled with the same compiler. With Ubuntu 16.04 and gcc-6, g++-6, clang-3.8 you should use these commands: if we use upgraded version of Ubuntu remove the versions prescribed above use it.

* Install HarfBuzz from package

HafBuzz is an Opentype text shaping engine.

* Verify that Mapnik has been correctly installed
* Install Apache HTTP Server

The Apache free open source HTTP server is among the most popular

web servers in the world.

* Install Python latest version
* Install openstreetmap-carto

For styling purpose of map, it should be installed alongside.

* Install font packs

Currently Noto fonts are used.

* Install Node.js
* Install PostgresSQL and PostGIS

PostgresSQL is the database used here and PostGIS is its spatial extender. It allow us to store geographic objects or map data in it. PostgresSQL+PostGIS are used for a wide variety of features such as rendering maps, geocoding, and analysis.

* Install Osm2pgsql

Osm2pgsql is an Openstreetmap specific software used to load the OSM data into the PostGIS database.

* Create the data folder

At least 20 GB HardDisk and appropriate RAM/SWAP is needed for this step (30 – 40 GB HD is better). 8 GB HD is not be enough with 1 GB RAM, configuring a swap is mandatory.

* Create Indexes and grant users

Create partial indexes to speed up the queries included in project.mml and grant access to all gis tables to avoid renderd errors when accessing tables with user tileserver.

* Configure rendered

Plug rendered and mod\_tile to the Apache webserver

* Configure Apache

Create a module load file.