OSGS

Release 0.1

Tim Sutton

Aug 09, 2021

CONTENTS:

1 Introduction **1**
- 1.1 Overview Diagram . . . 2

2 Preparing the server **3**
- 2.1 Basic Security . . . 3
- 2.2 Additional Software . . . 4

3 Initial Configuration **5**
- 3.1 User Group . . . 5
- 3.2 Project Checkout . . . 5
- 3.3 Fetching Docker Images . . . 6
- 3.4 Configuration . . . 6

4 Production Stack **7**
- 4.1 Overview . . . 7
- 4.2 Configuration . . . 8

5 SCP File Drop Service **9**
- 5.1 Directory layout for the QGIS projects folder . . . 11
- 5.2 Starting the container . . . 11
- 5.3 FAQ . . . 11

6 Postgres and PostGIS **13**
- 6.1 Overview . . . 13
- 6.2 Configuration . . . 13

7 PostgreSQL Workflows **15**
- 7.1 Direct connection . . . 15

8 QGIS Server Scaling **17**

9 GeoServer Configuration **19**

10 Working with Mergin projects **21**
- 10.1 Mergin-db-sync . . . 21
- 10.2 Mergin-client . . . 22

11 OSM Mirror **23**
- 11.1 PBF Container . . . 23
- 11.2 Clip Area . . . 23
- 11.3 Mappings . . . 24

11.4 Run services . . . 24
11.5 Publishing with GeoServer . . . 24
11.6 Publishing with QGIS Server . . . 27
11.7 OSM Attribution . . . 27

12 Obtaining free fonts for your projects **29**

13 Building Documentation **31**
13.1 HTML Docs . . . 31
13.2 PDF Docs . . . 31

14 Indices and tables **33**

CHAPTER

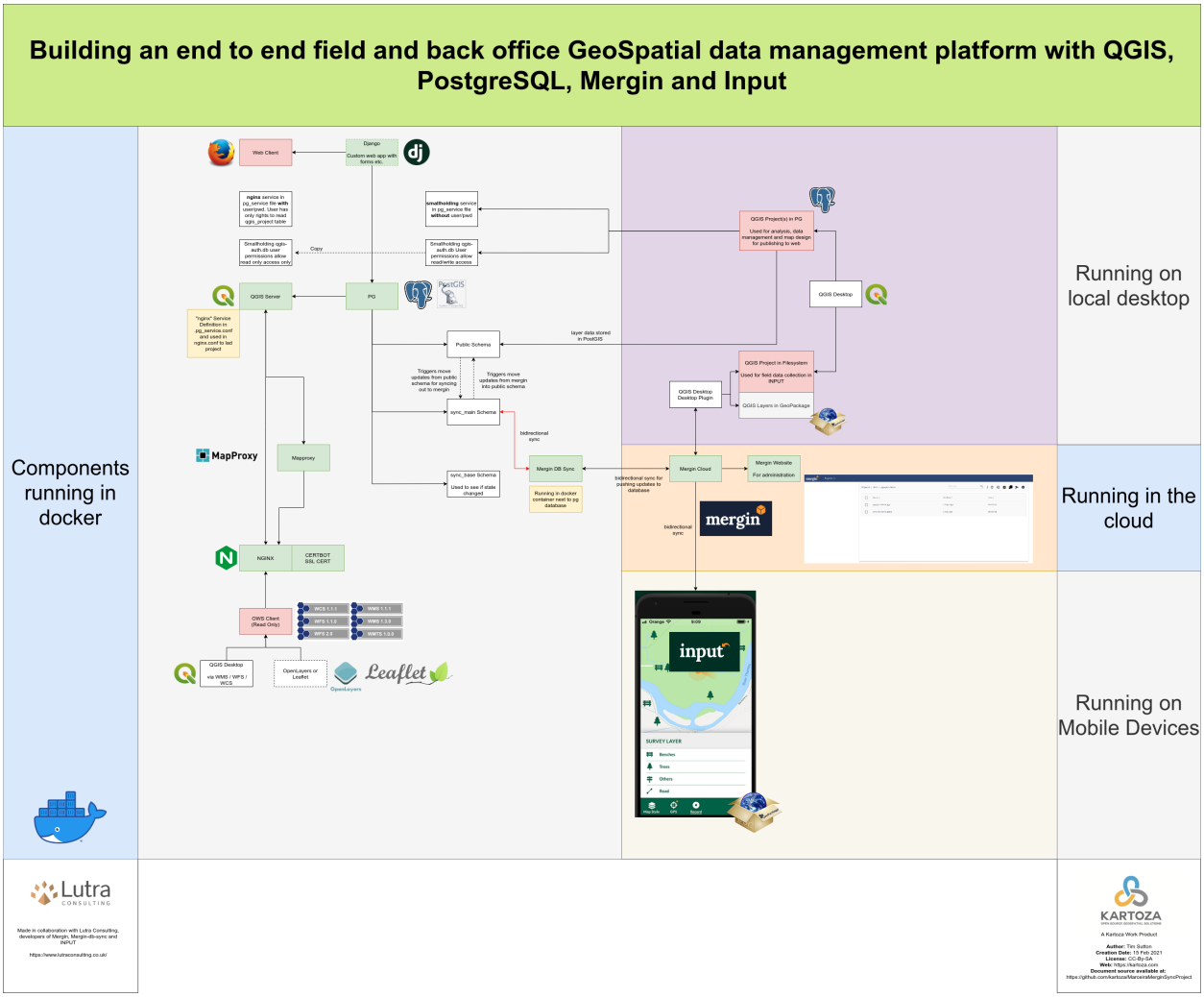
ONE

INTRODUCTION

This project provides a platform for creating, sharing and publishing data and maps using an Open Source GIS Stack. It has two primary objectives:

1. Provide a platform that integrates these technologies:
    1. One or more QGIS Projects
        1. Projects stored in-database in PostgreSQL
        2. Projects stored in the file system
    2. QGIS Server
        1. Publishing projects stored on the files system
        2. Publishing projects stored in the database
    3. Field data collection and project synchronisation support:
        1. The Mergin platform for cloud storage of projects
        2. The INPUT mobile data collection platform
        3. The Mergin-db-sync tool that will synchronise a Mergin cloud project into a PostgreSQL project.
    4. PostgreSQL and PostGIS running in Docker and providing a database backend for our stack.
    5. NGINX a lightweight web server acting as a proxy in front of QGIS server and as a server for the static HTML content.
    6. QGIS Server to serve QGIS projects from the database and from the file system.
    7. QGIS Server Docker Image from OpenQuake.
    8. Apache Superset to provide dashboard visualisations

1.1 Overview Diagram



CHAPTER

TWO

PREPARING THE SERVER

2.1 Basic Security

2.1.1 Unattended upgrades

This will automatically install only security fixes on a continual basis on your server.

```
sudo apt install unattended-upgrades
```

2.1.2 ssh

Disable password authentication for SSH

```
sudo vim /etc/ssh/sshd_config
```

Set this:

```
PasswordAuthentication no
Then do
sudo systemctl restart sshd.service
```

2.1.3 Crowdsec

https://crowdsec.net/

```
wget -qO - https://s3-eu-west-1.amazonaws.com/crowdsec.debian.pragmatic/crowdsec.asc␣
↪|sudo apt-key add - && echo "deb https://s3-eu-west-1.amazonaws.com/crowdsec.debian.
↪pragmatic/$(lsb_release -cs) $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.
↪d/crowdsec.list > /dev/null;
sudo apt-get update
sudo apt-get install crowdsec
```

2.1.4 Fail2ban

```
sudo apt install fail2ban
https://www.fail2ban.org/wiki/index.php/Main_Page
```

2.1.5 Firewall

```
sudo ufw allow ssh
sudo ufw enable
sudo ufw status
```

Should show something like this:

```
Status: active

To              Action   From
--              ------   ----
22/tcp            ALLOW    Anywhere
22/tcp (v6)         ALLOW    Anywhere (v6)
```

We will open more ports as they are needed.

2.1.6 Status monitoring

bpytop is a great console based dashboard for monitoring your server.

```
sudo snap install bpytop
```

2.2 Additional Software

2.2.1 Docker

```
sudo apt install docker.io docker-compose
```

2.2.2 Git, rpl, pwgen and Make

Needed for checking out our docker project and running the various make commands we provide.

```
sudo apt install git make rpl pwgen
```

CHAPTER

THREE

INITIAL CONFIGURATION

Note for the unprivileged user throughout here, we use the user name 'timlinux' in various examples - you should substitute this with your own user.

3.1 User Group

Add yourself to the user group of docker so you don't need to sudo docker commands.

```
sudo usermod -a -G docker timlinux
```

Then log out and in again to assume the upgraded permissions.

3.2 Project Checkout

```
cd /home
sudo mkdir web
sudo chown timlinux.timlinux web
cd web
git clone https://github.com/kartoza/OpenSource-GIS-Stack
cd OpenSource-GIS-Stack
```

3.3 Fetching Docker Images

```
[timlinux@fedora OpenSource-GIS-Stack]$ docker image ls
REPOSITORY                      TAG                 IMAGE ID       CREATED         SIZE
apache/superset                 latest-dev          dae66d8ef922   13 hours ago    1.9GB
pbf                             stable              24a6ed68e359   14 hours ago    252MB
swaggerapi/swagger-ui           latest              be6494c7edf4   32 hours ago    98.7MB
lutraconsulting/mergin-db-sync  latest              2f479ae94f5d   2 days ago      630MB
opendronemap/odm                latest              223a72b0d4e5   3 days ago      1.62GB
byrnedo/alpine-curl             latest              5208e5295614   9 days ago      6.9MB
redis                           latest              739b59b96069   10 days ago     105MB
nginx                           latest              62d49f9bab67   2 weeks ago     133MB
node                            14                  d6602e31594f   2 weeks ago     943MB
kartoza/docker-osm              imposm-latest       3d8739408ed0   3 weeks ago     1.08GB
kartoza/docker-osm              osmupdate-latest    bb1457cbd311   3 weeks ago     250MB
certbot/certbot                 latest              c8eac9ed295e   3 weeks ago     357MB
silexlabs/silex                 latest              be2ad993c076   4 weeks ago     1.38GB
klakegg/hugo                    0.82.0              2feec974b2db   5 weeks ago     46.1MB
openquake/qgis-server           stable              45d3c20d6088   2 months ago    1.26GB
kartoza/mapproxy                latest              d6df8e43db21   3 months ago    1.37GB
kartoza/postgis                 13.0                a85a969d604b   5 months ago    1.34GB
kartoza/geoserver               2.18.0              f6774787e652   6 months ago    1.88GB
postgrest/postgrest             latest              befe6cd943da   11 months ago   84.6MB
kartoza/docker-osm              osmenrich-latest    ccab39593491   22 months ago   949MB
jguyomard/hugo-builder          latest              650ec6415cde   2 years ago     57.2MB
quay.io/lkiesow/docker-scp      latest              b99cf24fe937   2 years ago     12.5MB
bytemark/webdav                 latest              c124350447bb   2 years ago     95.6MB
geodata/gdal                    latest              1e1929f80f44   3 years ago     1.29GB
```

3.4 Configuration

Copy the .env boilerplate file and then adjust any settings in it as needed.

```
cp .env.example .env
```

Replace terms that should be unique for your installation:

```
rpl example.org geoservices.govt.lc .env
rpl example.org geoservices.govt.lc nginx_conf/nginx.conf
```

CHAPTER

FOUR

PRODUCTION STACK

4.1 Overview

In this section we will bring up the full production stack, but to do that we first need to get an SSL certificate issued. To facilitate this, there is a special, simplified, version of Nginx which has no reverse proxies in place and not docker dependencies. Here is an overview of the process:

1. Replace the domain name in your letsencrypt init script
2. Replace the email address in your letsencrypt init script
3. Replace the domain name in the certbot init nginx config file
4. Open up ports 80 and 443 on your firewall
5. Run the init script, ensuring it completed successfully
6. Shut down the minimal nginx
7. Replace the domain name in the production nginx config file
8. Generate passwords for geoserver, postgres, postgrest and update .env
9. Copy over the mapproxy template files
10. Run the production profile in docker compose

At the end of the process you should have a fully running production stack with these services:

IMAGE	PORTS	NAMES
kartoza/mapproxy	8080/tcp	osgisstack_mapproxy_1
nginx	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	osgisstack_nginx_1
swaggerapi/swagger-ui	80/tcp, 8080/tcp	osgisstack_swagger_1
postgrest/postgrest	3000/tcp	osgisstack_postgrest_1
kartoza/geoserver:2.18.0	8080/tcp, 8443/tcp	osgisstack_geoserver_1
openquake/qgis-server:stable	80/tcp, 9993/tcp	osgisstack_qgis-server_1
kartoza/postgis:13.0	0.0.0.0:5432->5432/tcp	osgisstack_db_1
quay.io/lkiesow/docker-scp	0.0.0.0:2222->22/tcp	osgisstack_scp_1
certbot/certbot	80/tcp, 443/tcp	osgisstack_certbot_1

The following ports will be accessible on the host to the docker services. You can, on a case by case basis, allow these through your firewall using ufw (uncomplicated firewall) to make them publicly accessible:

1. 80 - http:Only really needed during initial setup of your letsencrypt certificate
2. 443 - https: All web based services run through this port so that they are encrypted

3. 5432 - postgres: Only expose this publicly if you intend to allow remote clients to access the postgres database.
4. 2222 - scp: The is an scp/sftp upload mechanism to mobilise data and resources to the web site

For those services that are not exposed to the host, they are generally made available over 443/SSL via reverse proxy in the Nginx configuration.

Some things should still be configured manually and deployed after the initial deployment:

1. Mapproxy configuration
2. setup.sql (especially needed if you are planning to use postgrest)
3. Hugo content management
4. Landing page static HTML

And some services are not intended to be used as long running services. especially the ODM related services.

4.2 Configuration

We have written a make target that automates steps 1-10 described in the overview above. It will ask you for your domain name, legitimate email address and then go ahead and copy the templates over, replace placeholder domain names and email address, generate passwords for postgres etc. and then run the production stack. Remember you need to have ufw, rpl, make and pwgen installed before running this command:

```
make configure
```

CHAPTER

FIVE

SCP FILE DROP SERVICE

This is a container intended for users to upload files for publication on the server. It runs on port 2222 so we need to expose that through the firewall:

```
sudo ufw allow 2222
```

You can add your public keys from the host e.g.

```
cat ~/.ssh/authorized_keys > scp_conf/gis_projects
```

Or copy them in by other means. Each file you create in scp_conf will be a user name when the scp container runs, with it's own directory in the storage volume, unless an explicit storage volume has been pre-defined (see list of these below). Each file should contain a list of public keys. If you add a key at some point, or a new user file, you may need to restart the container:

```
docker-compose profile=scp restart
```

The following scp shares are made for the various purposes listed below. You need to follow the same pattern of creating a config file for each. These shares each have a dedicated volume associated with it which is also mounted into the associated server container.

- **User:** geoserver_data
- **Named Volume:** scp_geoserver_data
- **Volume Mounted To:** scp, geoserver
- **Notes:** Copy vector and raster datasets here for publishing in GeoServer.
- **Example Use:** `sftp://geoserver_data@<hostname>:2222/home/geoserver_data`

- **User:** qgis_projects
- **Named Volume:** scp_qgis_projects
- **Volume Mounted To:** scp, qgis-server
- **Notes:** Copy QGIS projects and data here for publishing with QGIS Server. See notes on directory layout below.
- **Example Use:** `sftp://qgis_projects@<hostname>:2222/home/qgis_projects`

- **User:** qgis_svgs
- **Named Volume:** scp_qgis_svgs

- **Volume Mounted To:** scp, qgis-server
- **Notes:** Embed SVGs in styles by preference in QGIS. Use this drop if you have no way to use embeded SVGS.
- **Example Use:** ``sftp://qgis_svgs@:2222/home/qgis_svgs`

- **User:** qgis_fonts
- **Named Volume:** scp_qgis_fonts
- **Volume Mounted To:** scp, qgis-server
- **Notes:** Copy fonts directly into the root folder.
- **Example Use:** `sftp://qgis_fonts@<hostname>:2222/home/qgis_fonts`

- **User:** hugo_data
- **Named Volume:** scp_hugo_data
- **Volume Mounted To:** scp, hugo*
- **Notes:** Upload markdown files for static site generation with Hugo.
- **Example Use:** `sftp://hugo_data@<hostname>:2222/home/hugo_data`

- **User:** odm_data
- **Named Volume:** scp_odm_data
- **Volume Mounted To:** scp, odm *
- **Notes:** Upload imagery data for processing with ODM
- **Example Use:** `sftp://odm_data@<hostname>:2222/home/odm_data`

- **User:** general_data
- **Named Volume:** scp_general_data
- **Volume Mounted To:** scp
- **Notes:** General sharing directory. Later we will publish this under nginx for public downloads. Don't put any sensitive data in here.
- **Example Use:** `sftp://general_data@<hostname>:2222/home/general_data`

Note: Any user connecting to any of these shares will be able to see all other files from all other users. They will only have write access to the folder they are connecting to, for all other shares their access will be read only. If you want to further partition the access to files you can create multiple scp services, each with one of the mount points listed above. In so doing users would not be able to see the other mount points listed above.

5.1 Directory layout for the QGIS projects folder

When adding projects to the qgis_projects folder, you need to follow this convention strictly for the projects to be recognised by QGIS Server:

`qgis_projects/<project_name>/<project_name>.qgs`

For example:

`qgis_projects/terrain/terrain.qgs`

There is a convenience Make target that will copy your .ssh/authorized_keys file contents into each of the scp_config user files listed in the table above.

```
make setup-scp
```

5.2 Starting the container

```
docker-compose --profile=scp up -d scp
```

Example copying of data into the container from the command line:

```
scp -P 2222 sample-document.txt localhost:/data//gis_projects/gis_projects/gis_projects
```

In Nautilus (file manager in Linux Gnome Desktop) you can test by connecting

`sftp://<hostname>:2222/data/gis_projects`

into the red highlighted box below:

XXXXXXXXXXXXXXXXXXXX

After that open a second window and you can drag and drop files too and from the folder. Windows users can use the free WinSCP application to copy files to the server.

5.3 FAQ

Q: When connecting I get "Host key validation failure" or similar **A:** Remove the entry for the server in your ~/.ssh/known_hosts

CHAPTER

SIX

POSTGRES AND POSTGIS

6.1 Overview

We use the Kartoza PostGIS docker image available here: https://hub.docker.com/r/kartoza/postgis/

The project home page is here: https://github.com/kartoza/docker-postgis

The project includes detailed documentation so this section only contains details relevant to the Open Source GIS Stack configuration.

6.2 Configuration

6.2.1 Database password:

Generate a strong password:

```
pwgen 20 1
```

Replace the default docker password for the postgres user with the strong password:

```
rpl "POSTGRES_PASSWORD=docker" "POSTGRES_PASSWORD=<strong password>" .env
```

6.2.2 Service file configuration:

Service files entries serve two scenarios:

1. They are needed for opening QGIS projects stored in postgres with PG connection URI because at the project URI you cannot use QGIS authdb. If you prefer to store your projects on the file system, you should rather remove these lines (whole nginx section) since the authentication from pg_conf/pg_service.conf can be done more securely by QGIS authdb.
2. Used by your QGIS Server projects to connect to the database once the project is opened from either the file system of the database. You can either specify your password and username in service file or for more advanced configuration you can store user / password credentials in a QGIS authdb file. Refer to the authdb section and in qgis_conf/qgis-auth.db and the readme in that folder.

On your local machine you should create your own service file with the same service name but connection details that make sense when using the database from your local machine. When you upload your projects into the stack they will connect using the settings from the server hosted service file below assuming you used the same service name.

To carry out the service file configuration, copy, rename then edit the pg_service file in pg_config as per the example below (note that we also substitute in the database password created in the steps above).

```
cp pg_conf/pg_service.conf.example \ pg_conf/pg_service.confpassword=docker
rpl password=<your password> pg_conf/pg_service.conf
```

6.2.3 Deployment

```
docker-compose --profile=postgres up -d
```

Note that the default configuration opens the postgresql service to all hosts. This is a potential security hole. If you open the port on the firewall e.g.

```
ufw allow 5432 tcp
```

Then be sure to connect from pg clients like psql or QGIS with SSL enabled so that passwords and data are not transmitted in clear text.

6.2.4 Validation

Create a local pg_service.conf file like the example below and save it in `~/.pg_service.conf` or similar as appropriate to your operating system (see https://www.postgresql.org/docs/12/libpq-pgservice.html for details on configuration options).

```
[os-gis-stack]
dbname=gis
port=5432
host=<hostname of your server>
user=<your password>
password=docker
```

Now pass the server parameter to psql and list the databases as per the example below:

```
[timlinux@fedora ~]$ psql service=os-gis-stack -l
List of databases
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
gis	docker	UTF8	C.UTF-8	C.UTF-8	
postgres	postgres	UTF8	C.UTF-8	C.UTF-8	
template0	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres
template1	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres

```
(4 rows)
```

Test from QGIS is similar:

XXXXXXXXXXXXXX

Note that there was no need to supply any credentials other than the service file name.

CHAPTER

SEVEN

POSTGRESQL WORKFLOWS

There are a number of different workflows we will explore here:

- Connecting to the PostgreSQL database from your QGIS Desktop application
- Connecting to the PostgreSQL database from QGIS Server
- Connecting to the PostgreSQL database from GeoServer
- Connecting from other applications

Before we dive into the details here, let us quickly examine some basic use cases:

- You collect data in the field using Input and use the mergin-db-sync workflow to synchronised field collected data into the database, then publish maps for this data in QGIS Server.
- You want to publish a layer in GeoServer, so you connect to the database from your desktop, drag and drop a local file system layer into the database using QGIS, then publish the layer from GeoServer.
- You want to create a project in QGIS desktop that uses PostgreSQL for data storage, then publish that data as a QGIS Project.

There are many other workflows that the Open GIS Stack supports, it is really up to your imagination! What is key to understand are the mechanisms that you can use to connect to the database in different contexts. And so we will focus this section on the different connection modalities.

7.1 Direct connection

Direct connection to the server over the standard PostgreSQL port exposed to the public internet is probably the least secure but most convenient approach. We do not recommend this approach, and if you do follow it be sure to use the option to force remote clients to connect using SSL (and expect a performance penalty in the process).

Note also that connecting to a PostgreSQL database using QGIS from a remote host may often be slow and irritating to use on a daily basis, especially if your internet connection is not very fast and your datasets are large.

By default the docker-compose.yaml

Publishing with QGIS Server

The workflows described in this section apply equally to any database hosted

Further reading for understanding authentication with PostgreSQL using cert based authentication here:

https://joelonsql.com/2013/04/27/securing-postgresql-using-hostssl-cert-clientcert1/

CHAPTER

EIGHT

QGIS SERVER SCALING

If your server has the needed resources, you can dramatically improve response times for concurrent QGIS server requests by scaling the QGIS server:

```
docker-compose --profile=production up -d --scale qgis-server=10 --remove-orphans
```

To take advantage of this, the locations/upstreams/qgis-server.conf should have one server reference per instance e.g.

```
    upstream qgis-fcgi {
        # When not using 'host' network these must reflect the number
        # of containers spawned by docker-compose and must also have
        # names generated by it (including the name of the stack)
        server osgisstack_qgis-server_1:9993;
        server osgisstack_qgis-server_2:9993;
        server osgisstack_qgis-server_3:9993;
        server osgisstack_qgis-server_4:9993;
        server osgisstack_qgis-server_5:9993;
        server osgisstack_qgis-server_6:9993;
        server osgisstack_qgis-server_7:9993;
        server osgisstack_qgis-server_8:9993;
        server osgisstack_qgis-server_9:9993;
        server osgisstack_qgis-server_10:9993;
    }
```

Then restart Nginx too:

```
docker-compose --profile=production restart nginx
```

Note that if you do an Nginx up it may bring down your scaled QGIS containers so take care.

Finally check the logs of Nginx to make sure things are running right:

```
docker-compose --profile=production logs nginx
```

CHAPTER

NINE

GEOSERVER CONFIGURATION

Remind user to set the master password as per https://docs.geoserver.geo-solutions.it/edu/en/security/security_overview.html#the-master-password (which is different to the admin password).

CHAPTER

TEN

WORKING WITH MERGIN PROJECTS

There are two modalities in which you can work with Mergin projects:

1. **mergin-db-sync**: A Mergin project which is synchronised into a PostgreSQL database and supports bidirectional syncing and editing.
2. **mergin-client:** A folder containing multiple mergin projects (all of the projects shared with a mergin user). These projects are synchronised into the filesystem and published via QGIS Server as web mapping services.

10.1 Mergin-db-sync

In the .env file you should specify these options:

- **MERGIN_USER:** This is the user account that will be used to log in and pull/push updates to the Mergin project.
- **MERGIN_PASSWORD:** This is the user password for the above account.
- **MERGIN_PROJECT_NAME:** Specified in the form of `user/project` this is the Mergin project that will be synchronised into the database.
- **MERGIN_SYNC_FILE:** This is the name of a GeoPackage `yourgeopackage.gpkg` in the Mergin project whose schema will be replicated into a PostGIS schema as described below.
- **DB_SCHEMA_MODIFIED:** This is a PostgreSQL schema (schemas can be thought of as 'folders' within your database within which tables are found) that will contain the synchronised data form mergin. The content of the tables are editable via INSERT/UPDATE/DELETE operations, bit the structure of these tables (via ALTER commands) should not be attempted. Note that the replication is bidirectional, so changes made in the database are synchronised to all mergin clients and changes made in the distributed clients will make their way back into your database.
- **DB_SCHEMA_BASE:** This is a 'hands off' copy of the content in the MERGIN_SYNC_FILE that is stored in PostgreSQL to act as a reference when mergin calculates the changeset between the MODIFIED schema content and the remote copies of the data. DO NOT USE THIS and definately DO NOT EDIT THIS..
- **MERGIN_URL:** This is the public server where your mergin project is hosted. By default it would be "https://public.cloudmergin.com" unless you are self hosting the Mergin backend, or using an alternative hosted instance.

Note that in the docker-compose file, the assumption is made that the database being used for Mergin syncing is called 'gis' and the hostname (in the private docker network) is called 'db'. The username and password are taken from the following keys in the .env file:

- POSTGRES_USER
- POSTGRES_PASSWORD

10.2 Mergin-client

In the second situation, you use the Mergin client to clone one of more Mergin projects to the host running docker-compose and these projects are made available through QGIS Server.

One critical note is that the Project directory and the Project File names must be the same, otherwise QGIS Server will not recognise the project as being valid. For example:

- Valid: `FooProject/FooProject.qgz`
- Not Valid: `FooProject/BarProject.qgz`

Once published in this way, valid projects will be accessible from any OGC compliant client (e.g. QGIS Desktop, OpenLayers, Leaflet) using the following URL Scheme:

`https://yourhost.com/ogc/yourproject`

For example, here is one we published for a client (domain name changed):

`https://example.org/ogc/Elevation`

You can read more about the mergin-client at the separate git repo here:

https://github.com/kartoza/mergin-client

CHAPTER

ELEVEN

OSM MIRROR

The OSM mirror uses the kartoza/docker-osm tool to create an in-database mirror of a designated geographical area in the designated postgres databse schema (set to: osm). The OSM mirror tool is described in the project README here:

https://github.com/kartoza/docker-osm

To deploy the docker-mirror you need to follow the steps descibed below. First a process overview:

1. Create the PBF feature container passing it a URL to a PBF file
2. Create a clip file that will be used to constrain any retrieved / imported data to a specific geographic area.
3. Tweak the mappings.yml file (advanced users)
4. Run the docker-osm service
5. Optionally include these data in published services via QGIS projects, GeoServer etc.

11.1 PBF Container

During the `make configure` process, the script will ask for the URL to an OSM .pbf file e.g.:

```
------------------------------------------------------------------
Fetching pbf if not cached and then copying to settings dir
------------------------------------------------------------------
URL For Country PBF File: https://download.geofabrik.de/central-america-latest.osm.pbf
```

You can enter any valid URL for an OSM .PBF file at this point. A docker container will be built that fetches the PBD and stores it on the host file system under osm_config.

11.2 Clip Area

Create the clip area to constrain the geographical region that data will be harvested for. For best performance, a simple rectangle is best, but any complext polygon can be used. The clip area must be saved as `osm_config/clip.geojson`. The format for the clip area must be GeoJSON. You can easily create this using QGIS.

11.3 Mappings

For advanced users, you can tweak the `osm_config/mapping.yml`

You can see how the imposm3 mapping syntax works here:

https://imposm.org/docs/imposm3/latest/mapping.html

Note that you cannot alter the mappings after the service is running without clearing the databse and restarting the import.

11.4 Run services

To run the OSM services do:

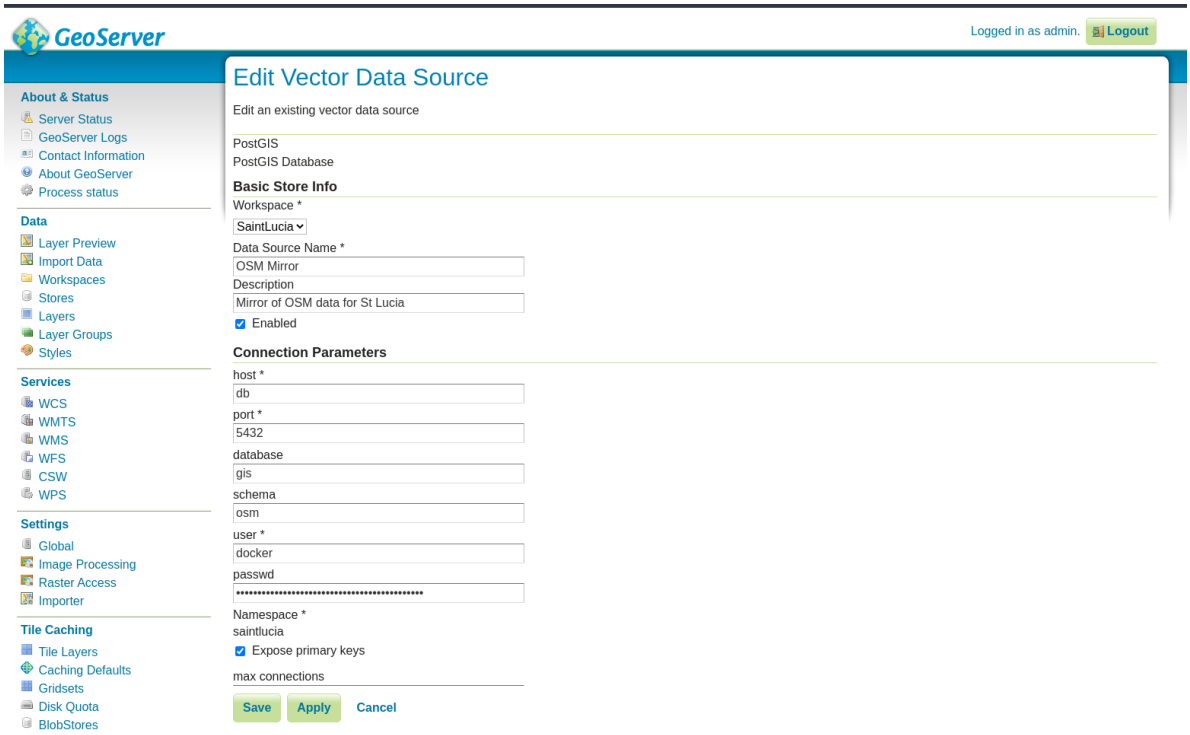
```
docker-compose --profile=osm up -d
```

11.5 Publishing with GeoServer

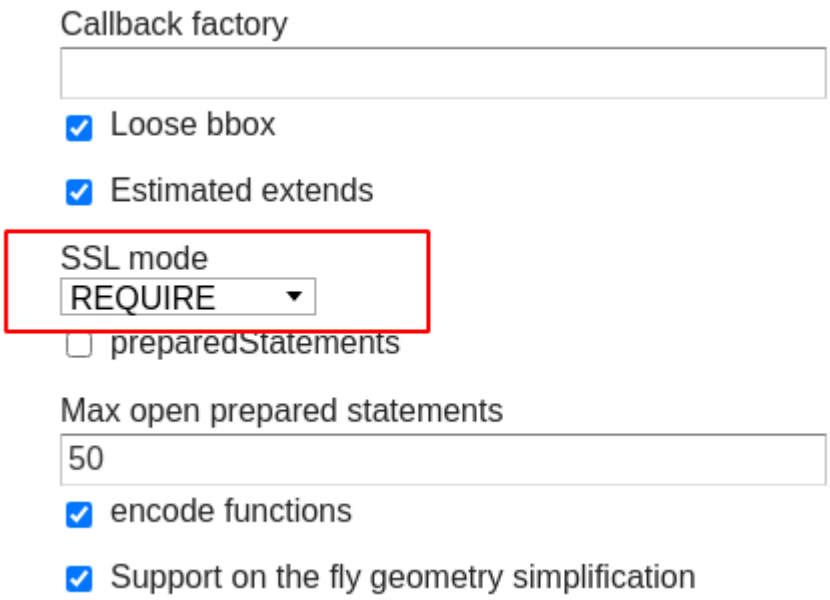
You can publish the data in the osm schema using GeoServer or by publishing a QGIS project that references the data layers in the OSM schema.

The steps for publishing with GeoServer are quite simple:

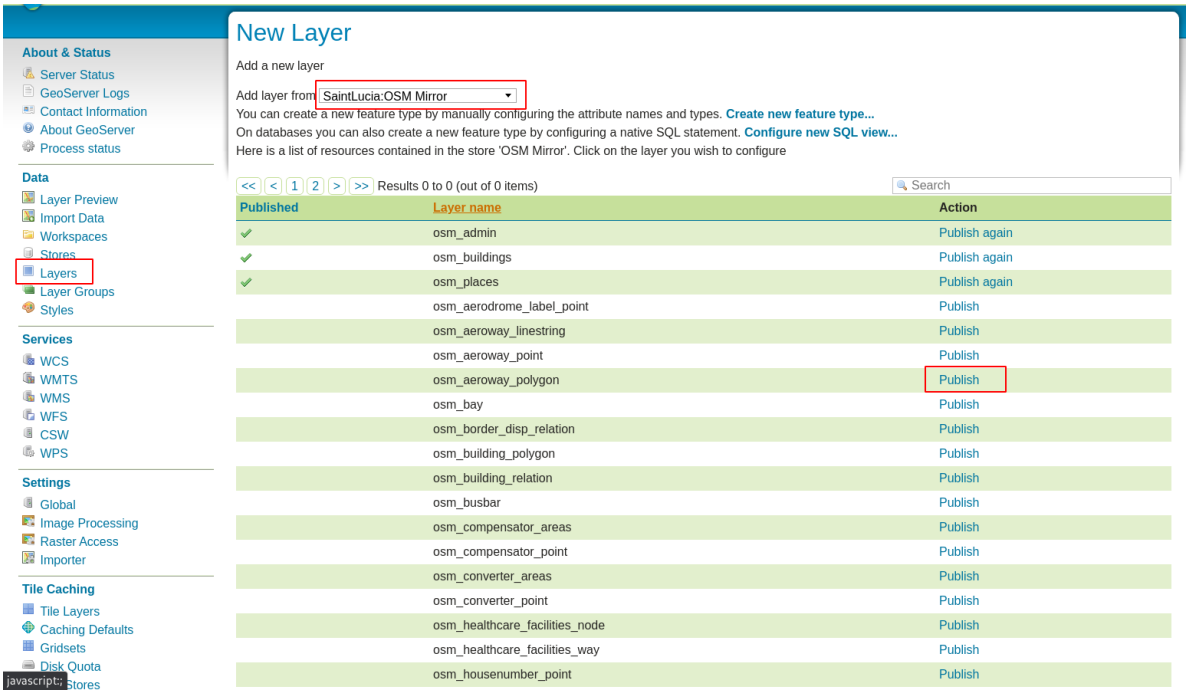
1. Log in to GeoServer using the 'admin' user and the password in .env.
2. Create a new store of type 'Postgis' and configure it as per the screenshot below, replacing the password with the Postgres password stored in .env:



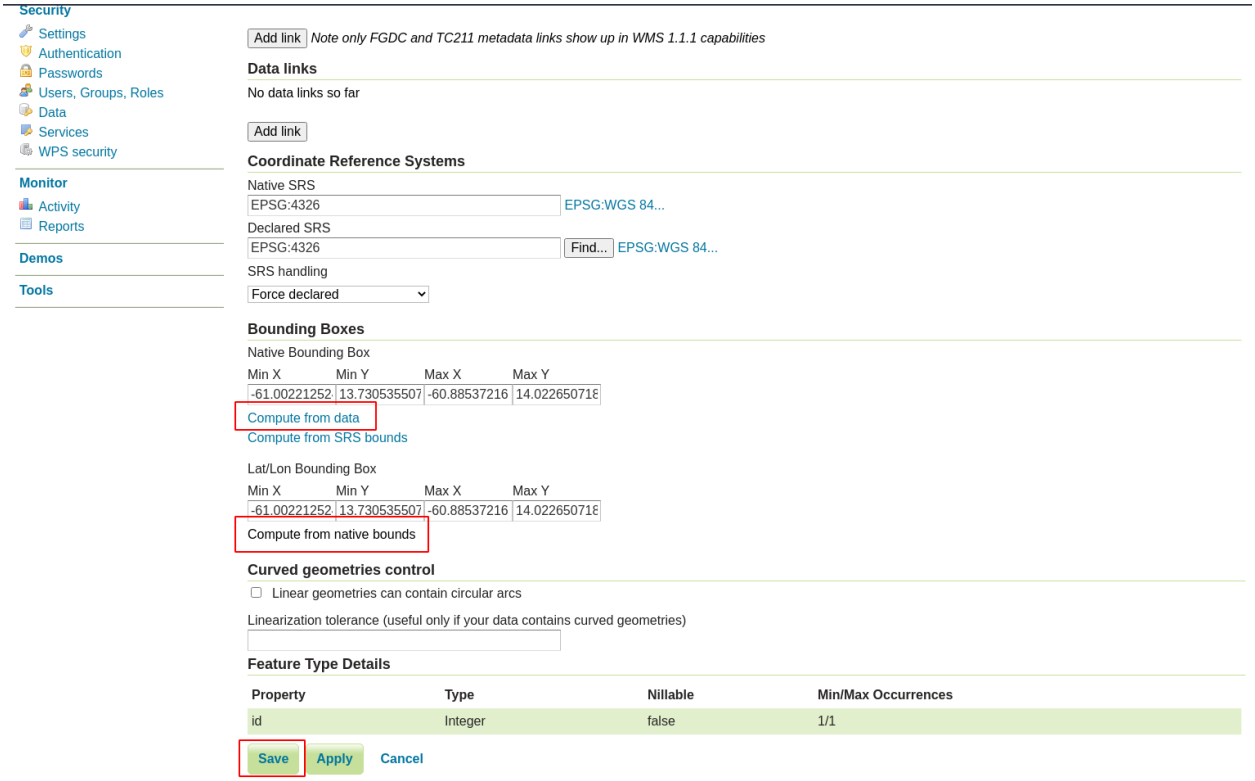
Also, be sure to scroll down and set SSL mode to Required:



3. Register one or more layers from that store as per the image below:



4.Complete the layer details as appropriate and make sure to click the options highlighted in red in the screenshot below:

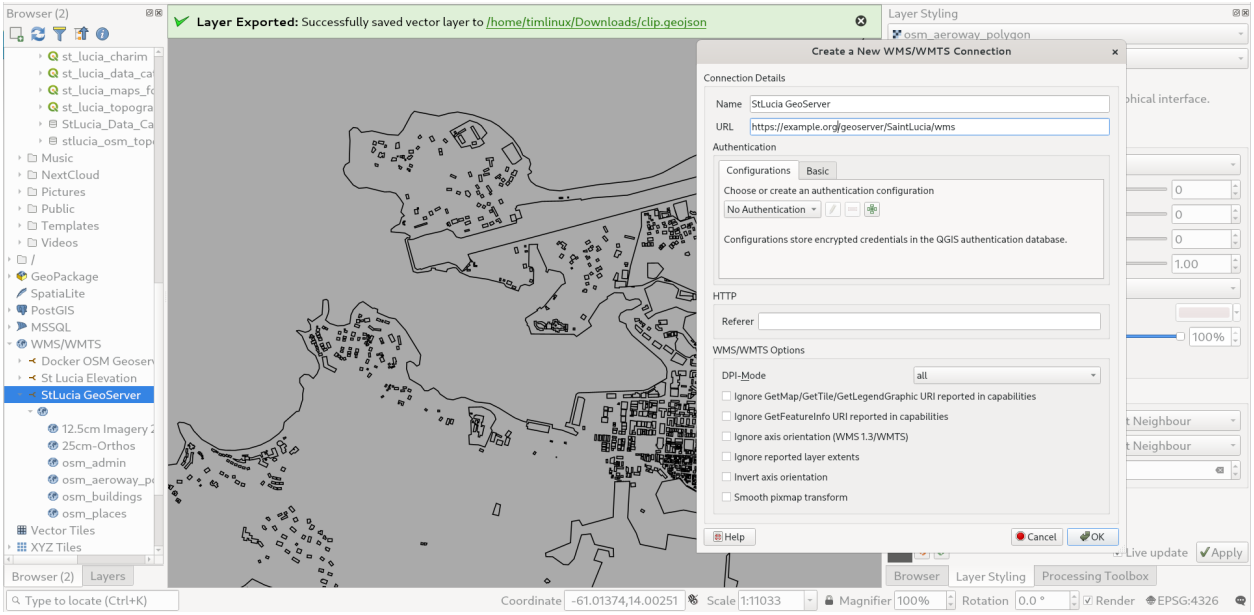


1. Connect to the GeoServer from a client e.g. QGIS using WFS or WMS using the scheme:

https://example.org/geoserver/SaintLucia/wfs

or

https://example.org/geoserver/SaintLucia/wms



11.6 Publishing with QGIS Server

The workflows described in the section on working with the PostgreSQL database below are basically all you need to know, so we don't repeat that here, other than to remind you that hte OSM mirrored data is by default stored in a schema called 'osm'.

11.7 OSM Attribution

Note that whenever you publish a map containing OSM data, be careful to adhere to the license and credit the OSM Project as per:

https://www.openstreetmap.org/copyright

CHAPTER

TWELVE

OBTAINING FREE FONTS FOR YOUR PROJECTS

There are two great sources of free fonts that you can use for your projects:

http://ftp.gnu.org/gnu/freefont/freefont-ttf-20120503.zip https://fonts.google.com/

There is a makefile target that can be run with:

```
make get-fonts
```

That will fetch all of these fonts for you. If you also fetch these fonts on your local machine and place them in your `~/.fonts` folder then you can use them in your local QGIS projects, know they will also be available to QGIS server if you publish that project as a web map.

Note: This and other makefile targets assume that you have not changed the `COMPOSE_PROJECT_NAME=osgisstack` environment variable in .env.

Note: The above make command fetches a rather large download!

CHAPTER

THIRTEEN

BUILDING DOCUMENTATION

13.1 HTML Docs

```
sudo dnf install sphinx
pip3 install --upgrade myst-parser
pip install sphinx-sizzle-theme
```

See https://sphinx-themes.org/sample-sites/sphinx-sizzle-theme/ for theme specific info.

See https://www.sphinx-doc.org/en/master/usage/markdown.html for Markdown in Sphinx support notes (and the docs at https://myst-parser.readthedocs.io/en/latest/syntax/optional.html). Especially, note the admonitions docs which are used to make little alert etc boxes in the docs: https://myst-parser.readthedocs.io/en/latest/syntax/optional.html#html-admonitions

13.2 PDF Docs

On fedora I just install the huge, full texlive package:

```
sudo dnf install texlive-scheme-full
```

Then build:

```
make latexpdf
```

Sometimes you need to run it a second time if it is a fresh build.

After the build is done, the PDF will be at:

```
docs/build/latex/osgs.pdf
```

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search