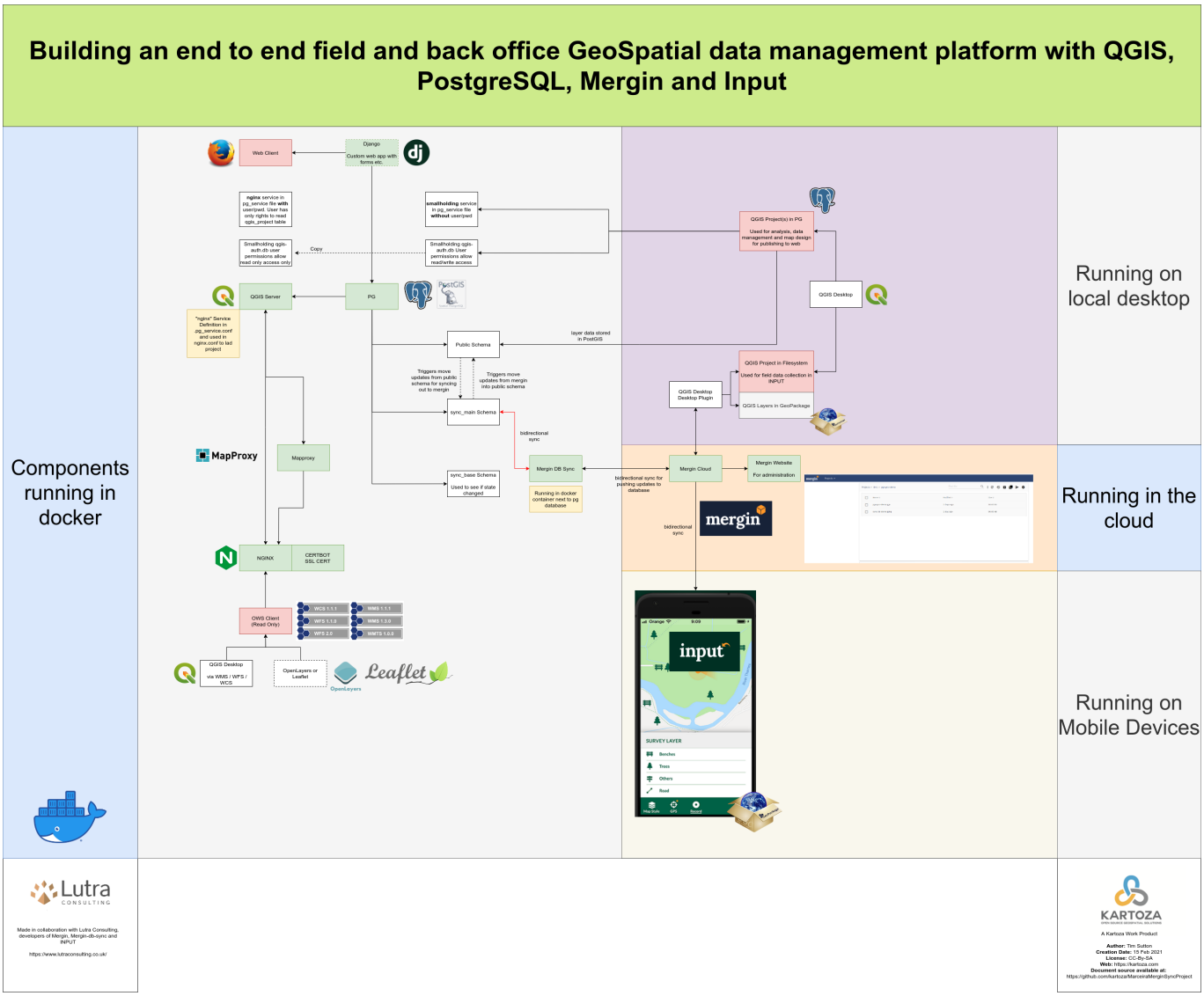
Introduction

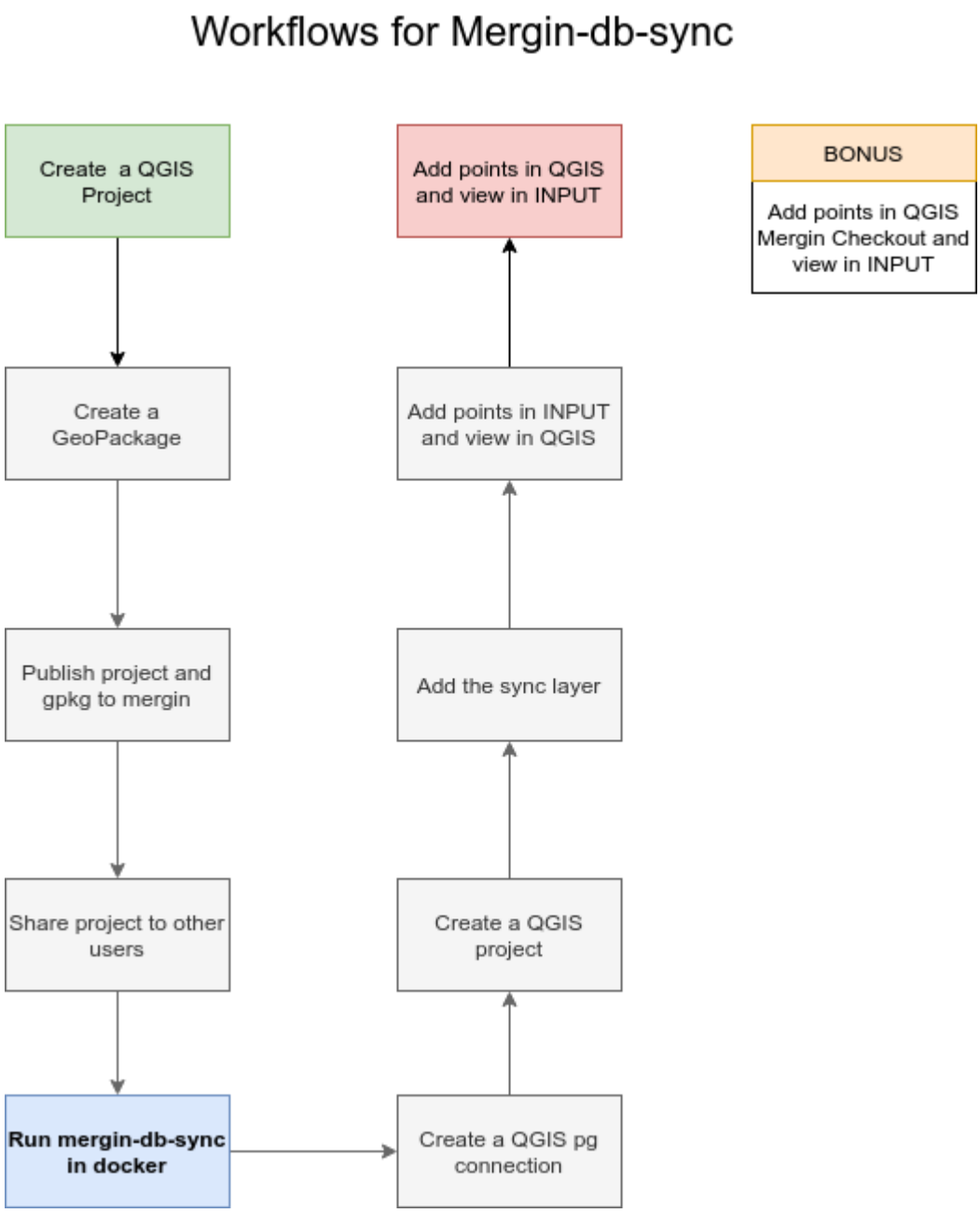
This project provides a platform for creating, sharing and publishing data and maps for your smallholding. It has two primary objectives:

1. Provide a demonstrator 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
    3. The Mergin platform
    4. The INPUT mobile data collection platform
    5. The Mergin-db-sync tool that will synchronise a Mergin cloud project into a PostgreSQL project.
    6. PostgreSQL and PostGIS running in Docker and providing a database backend for our stack.
    7. NGINX a lightweight web server acting as a proxy in front of QGIS server and as a server for the static HTML content.
    8. QGIS Server to serve QGIS projects from the database and from the file system.
    9. QGIS Server Docker Image from OpenQuake.

Overview Diagram



Generalised Workflow



Getting started

Define your domain name

This repo contains a worked example of running the stack as described above. There are numerous references to the testing domain 'castelo.kartoza.com' in various configuration files that should be replaced with your own preferred domain name before running any of these images. One simple way to do so is to install the 'rpl' command line tool and then replace all instances of the aforementioned domain named e.g.:

sudo apt install rpl rpl castelo.kartoza.com your.domain.com *

After doing that make sure you have a valid DNS entry pointing to your host - you will need this for the Certbot/Letsencrypt bot to work.

Similarly there is a reference to my email in the letsencrypt init script which you need to change to your own email address in `init-letsencrypt.sh`.

Initialise Certbot

Make sure the steps above have been carried out then run the init script.

```
./init-letsencrypt.sh
```

After successfully running it will terminate wiith a message like this:

```
### Requesting Let's Encrypt certificate for castelo.kartoza.com ...
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator webroot, Installer None
Requesting a certificate for castelo.kartoza.com
Performing the following challenges:
http-01 challenge for castelo.kartoza.com
Using the webroot path /var/www/certbot for all unmatched domains.
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/castelo.kartoza.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/castelo.kartoza.com/privkey.pem
   Your certificate will expire on 2021-05-30. To obtain a new or
   tweaked version of this certificate in the future, simply run
   certbot again. To non-interactively renew *all* of your
   certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le


### Reloading nginx ...
2021/03/01 22:50:52 [notice] 33#33: signal process started
```

If you have any issues checking out the certificate etc. then check the nginx logs:

```
docker-compose logs -f nginx
```

Check Services

After the above steps, a subset of the services will be running.

```
docker-compose ps
```

Which should show something like this:

```
             Name                           Command                  State
Ports
-------------------------------------------------------------------------------
```

```
-----------------------------------
maceiramergindbsync_db_1          /bin/sh -c /scripts/docker ...   Up
(healthy)   0.0.0.0:15432->5432/tcp
maceiramergindbsync_geoserver_    /bin/sh /scripts/entrypoint.sh   Up
(healthy)   8080/tcp, 8443/tcp
1
maceiramergindbsync_mapproxy_1    /start.sh mapproxy-util se ...   Up
8080/tcp
maceiramergindbsync_nginx_1       /docker-entrypoint.sh /bin ...   Up
0.0.0.0:443->443/tcp,

0.0.0.0:80->80/tcp
maceiramergindbsync_postgrest_    /bin/sh -c exec postgrest  ...   Up
0.0.0.0:32779->3000/tcp
1
maceiramergindbsync_qgis-         /bin/sh -c /usr/local/bin/ ...   Up
80/tcp,
server_1
0.0.0.0:32780->9993/tcp
maceiramergindbsync_swagger_1     /docker-entrypoint.sh sh / ...   Up
80/tcp, 0.0.0.0:3001->8080/tcp
```

Before we bring up the OSM mirror and the Mergin Sync serviice we need to do some additional configuration. In the next subsection we will set up the OSM mirror clip region:

Set Your Clip Region

Essential Reading

QGIS Server

You should read the QGIS Server documentation on QGIS.org. It is well written and covers a lot of background explanation which is not provided here. Also you should familiarise yourself with the Environment Variables.

Alesandro Passoti has made a number of great resources available for QGIS Server. See his workshop slide deck and his server side plugin examples, and more examples here.

PostgREST

Take special note of the fact that the passing of environment variables to the docker container is desribed here. Especially this line:

> These variables match the options shown in our Configuration section, except they are capitalized, have a PGRST_ prefix, and use underscores.

So for example `openapi-server-proxy-ur` would become `PGRST_OPENAPI_SERVER_PROXY_URI`.

This latter environment variable is important by the way to present a public url for the api running inside the docker container.

I based some of my Nginx configuration on the excellent example by Johnny Lambada.

Loading Raster Layers

Here is how I loaded raster data into the database:

Flags used:

```
-d Drop table, create new one and populate it with raster(s)
-t TILE_SIZE Cut raster into tiles to be inserted one per table row.
TILE_SIZE
   is expressed as WIDTHxHEIGHT or set to the value "auto" to allow the
loader to
   compute an appropriate tile size using the first raster and applied to
all
   rasters.
-F Add a column with the name of the file
-I Create a GiST index on the raster column.
-s <SRID> Assign output raster with specified SRID. If not provided or is
zero,
   raster's metadata will be checked to determine an appropriate SRID.
-l OVERVIEW_FACTOR Create overview of the raster. For more than one factor,
   separate with comma(,). Overview table name follows the pattern
o_overview
   factor_table, where overview factor is a placeholder for numerical
overview
   factor and table is replaced with the base table name. Created overview
is
   stored in the database and is not affected by -R. Note that your
generated sql
   file will contain both the main table and overview tables.
```

(Above copied directly from raster2pgsql help docs)

```
echo "create schema raster;" | psql -h localhost -p 15432 -U docker gis
cd /home/timlinux/gisdata/Maceira/orthophoto
raster2pgsql -s 32629 -t 256x256 -C -l 4,8,16,32,64,128,256,512 -P -F -I
odm_orthophoto.tif raster.orthophoto | psql -h localhost -p 15432 -U docker
gis
cd /home/timlinux/gisdata/Maceira/elevation
raster2pgsql -s 32629 -t 256x256 -C -l 4,8,16,32,64,128,256,512 -d -P -F -I
dtm.tif raster.dtm | psql -h localhost -p 15432 -U docker gis
raster2pgsql -s 32629 -t 256x256 -C -l 4,8,16,32,64,128,256,512 -d -P -F -I
dsm.tif raster.dsm | psql -h localhost -p 15432 -U docker gis
cd -
```

Authentication Management

Some discussion suggest to set authdb configuration parameters in Apache/Nginx but I found it would only work if I set these in the environment of the QGIS Server docker container.

Other Notes

Fonts

Fonts used in this project are from Google Fonts, checked out using:

```
git clone git@github.com:google/fonts.git
```