Setting up the Open Source GIS Stack

Tim Sutton

tim@kartoza.com

(c) Tim Sutton, Jan 2021

# Contents

[Setting up the Open Source GIS Stack 1](#_Toc1)

[1. Contents 1](#_Toc2)

[2. Preparing the server: 2](#_Toc3)

[1. Security tools: 2](#_Toc4)

[Unattended upgrades 2](#_Toc5)

[ssh 2](#_Toc6)

[Crowdsehttps://github.com/kartoza/docker-postgisc 2](#_Toc7)

[Fail2ban 3](#_Toc8)

[Firewall 3](#_Toc9)

[2. Status monitoring 3](#_Toc10)

[3. Docker 4](#_Toc11)

[Git, rpl and Make 4](#_Toc12)

[3. Deploying the server 4](#_Toc13)

[4. User Group 4](#_Toc14)

[5. Project Checkout 4](#_Toc15)

[6. Fetching Docker Images 5](#_Toc16)

[7. Configuration 5](#_Toc17)

[4. Services 5](#_Toc18)

[8. SCP File Drop 5](#_Toc19)

[1. Postgres and PostGIS 8](#_Toc20)

[Overview 8](#_Toc21)

[Configuration 8](#_Toc22)

[Deployment 8](#_Toc23)

# Preparing the server:

## Security tools:

### Unattended upgrades

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

sudo apt install unattended upgrades

### ssh

Disable password authentication for SSH

sudo vim /etc/ssh/sshd\_config

Set this:

PasswordAuthentication no

Then do

sudo systemctl restart sshd.service

### 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

### Fail2ban

sudo apt install fail2ban

<https://www.fail2ban.org/wiki/index.php/Main_Page>

### 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.

## Status monitoring

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

sudo snap install bpytop

## Docker

sudo apt install [docker.io](http://docker.io) docker-compose

### 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

# Deploying the server

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

## 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.

## Project Checkout

cd /home

sudo mkdir web

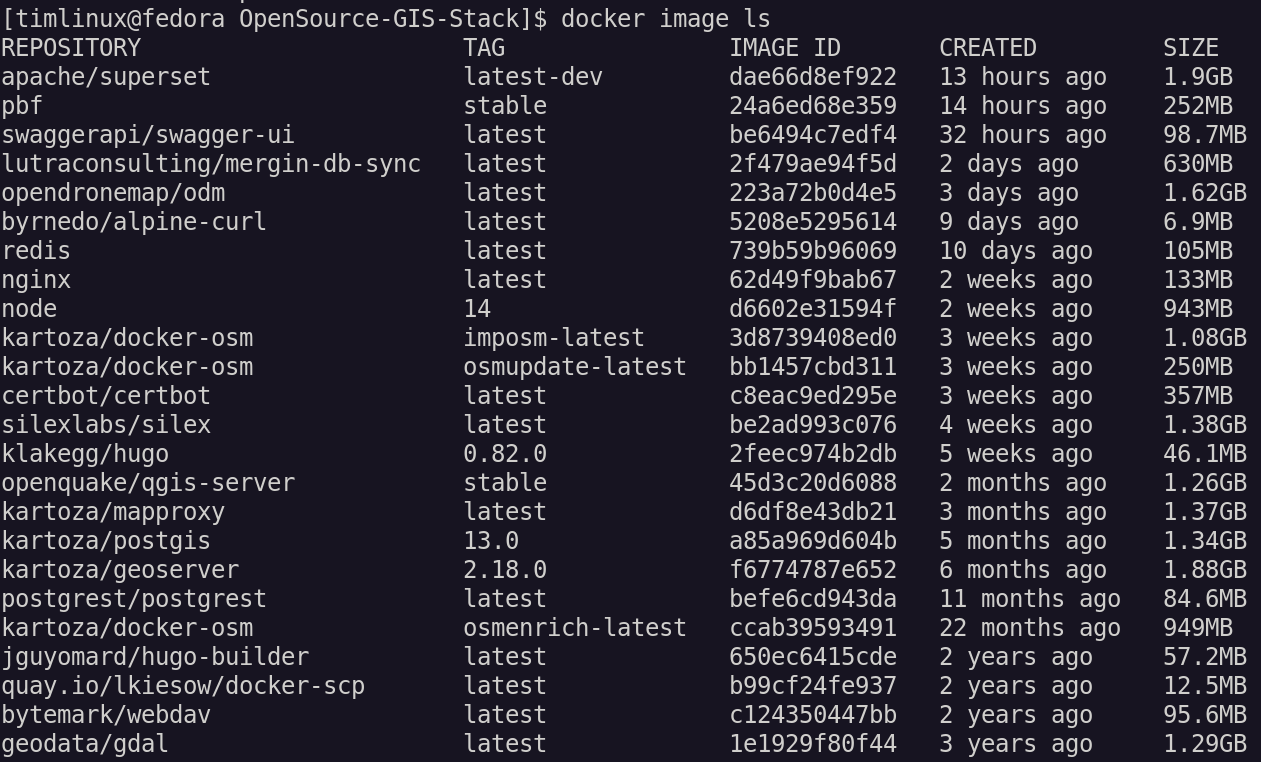
sudo chown [timlinux.timlinux](http://timlinux.timlinux) web

cd web

git clone https://github.com/kartoza/OpenSource-GIS-Stack

cd OpenSource-GIS-Stack

## Fetching Docker Images



## 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

# Services

## SCP File Drop

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 Key File in scp\_config** | **Named Volume** | **Mounted To** | **Notes** |
| geoserver\_data | scp\_geoserver\_data | scp, geoserver | Copy vector and raster datasets here for publishing in GeoServer. |
| sftp://geoserver\_data@<hostname>:2222/home/geoserver\_data | | | |
| qgis\_projects | scp\_qgis\_projects | scp, qgis-server | Copy QGIS projects and data here for publishing with QGIS Server. See notes on directory layout below. |
| sftp://qgis\_projects@<hostname>:2222/home/qgis\_projects | | | |
| qgis\_svgs | scp\_qgis\_svgs | scp, qgis-server | Embed SVGs in styles by preference. |
| sftp://qgis\_svgs@<hostname>:2222/home/qgis\_svgs | | | |
| qgis\_fonts | scp\_qgis\_fonts | scp, qgis-server | Copy fonts directly into the root folder. |
| sftp://qgis\_fonts@<hostname>:2222/home/qgis\_fonts | | | |
| hugo\_data | scp\_hugo\_data | scp, hugo\* | Upload markdown files for static site generation with Hugo |
| sftp://hugo\_data@<hostname>:2222/home/hugo\_data | | | |
| odm\_data | scp\_odm\_data | scp, odm \* | Upload imagery data for processing with ODM |
| sftp://odm\_data@<hostname>:2222/home/odm\_data | | | |
| general\_data | scp\_general\_data | scp | General sharing directory. Later we will publish this under nginx for public downloads. Don’t put any sensitive data in here. |
| 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.

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

Starting the container:

docker-compose up -d scp

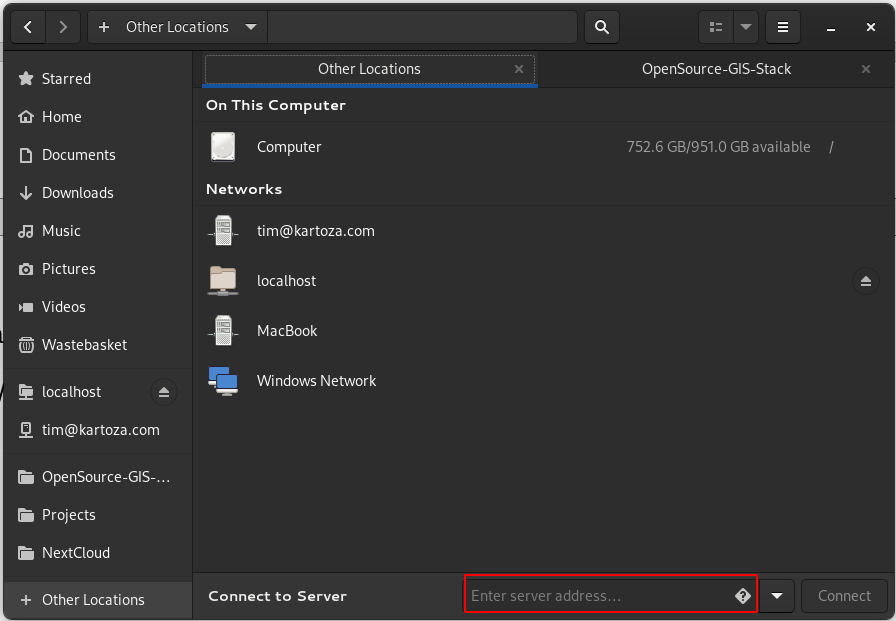
Example copying of data into the container from the command line:

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:



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.

### FAQ:

Q: When connecting I get “Host key validation failure” or similar

A: Remove the entry for the server in your ~/.ssh/known\_hosts

## Postgres and PostGIS

### 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.

### Configuration

**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

**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

### 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.

### Validation

Create a local [pg\_service.conf](http://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 +

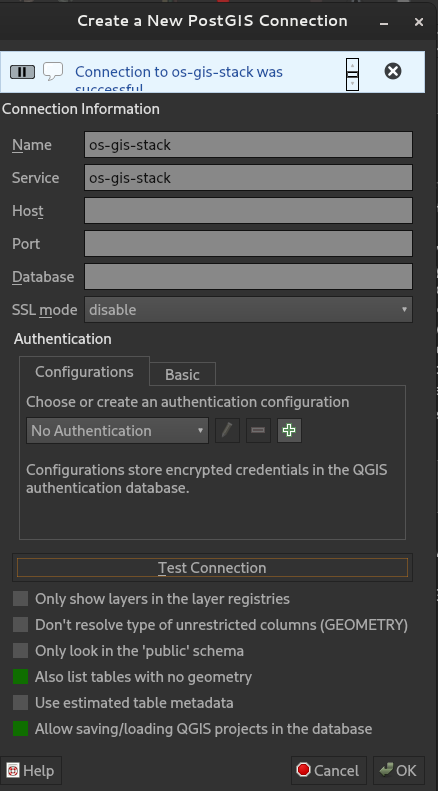
| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | C.UTF-8 | C.UTF-8 | =c/postgres +

| | | | | postgres=CTc/postgres

(4 rows)

Test from QGIS is similar:



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

## Production Stack

### 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](http://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.

### 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

# 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).