



QGIS Server 3.16 User Guide

QGIS Project

Mar 24, 2022

CONTENTS

1	Introduction	1
2	Getting Started	3
2.1	Installation on Debian-based systems	3
2.1.1	Apache HTTP Server	4
2.1.2	NGINX HTTP Server	6
2.1.3	Xvfb	11
2.2	Installation on Windows	12
2.3	Serve a project	14
2.4	Configure your project	15
2.4.1	WMS capabilities	17
2.4.2	WFS capabilities	18
2.4.3	WCS capabilities	18
2.4.4	Fine tuning your OWS	18
2.5	Integration with third parties	18
2.5.1	Integration with QGIS Desktop	18
2.5.2	Integration with MapProxy	18
2.5.3	Integration with QWC2	19
3	Services	21
3.1	Web Map Service (WMS)	21
3.1.1	GetMap	22
3.1.2	GetFeatureInfo	29
3.1.3	GetPrint	32
3.1.4	GetLegendGraphics	34
3.1.5	GetProjectSettings	35
3.2	Web Feature Service (WFS)	35
3.2.1	GetFeature	36
3.3	Web Map Tile Service (WMTS)	38
3.3.1	GetCapabilities	38
3.3.2	GetTile	39
3.3.3	GetFeatureInfo	41
3.4	WFS3 (OGC API Features)	43
3.4.1	Resource representation	43
3.4.2	Endpoints	44
3.4.3	Pagination	48
3.4.4	Feature filtering	48
3.4.5	Feature sorting	49
3.4.6	Attribute selection	49
3.4.7	Customize the HTML pages	50
3.5	Extra parameters supported by all request types	51
3.6	REDLINING	51
3.7	External WMS layers	53
3.8	QGIS Server catalog	53

4	Plugins	57
4.1	Installation	57
4.2	HTTP Server configuration	57
4.2.1	Apache	57
4.3	How to use a plugin	58
5	Advanced configuration	59
5.1	Logging	59
5.2	Environment variables	59
5.3	Settings summary	61
5.4	Short name for layers, groups and project	62
5.5	Connection to service file	62
5.6	Add fonts to your linux server	62
6	Development Server	65
7	Containerized deployment	67
7.1	Simple docker images	67
7.1.1	First run	68
7.1.2	Usable sample	69
7.1.3	Cleanup	70
7.2	Docker stacks	70
7.2.1	Swarm/docker-compose	70
7.2.2	Kubernetes	71
7.3	Cloud deployment	74
7.3.1	AWS usecase	75
8	Frequently Asked Question	77

INTRODUCTION

QGIS Server is an open source WMS, WFS, OGC API for Features 1.0 (WFS3) and WCS implementation that, in addition, implements advanced cartographic features for thematic mapping. QGIS Server is a FastCGI/CGI (Common Gateway Interface) application written in C++ that works together with a web server (e.g., Apache, Nginx). It has Python plugin support allowing for fast and efficient development and deployment of new features.

QGIS Server uses QGIS as back end for the GIS logic and for map rendering. Furthermore, the Qt library is used for graphics and for platform-independent C++ programming. In contrast to other WMS software, the QGIS Server uses cartographic rules as a configuration language, both for the server configuration and for the user-defined cartographic rules.

As QGIS desktop and QGIS Server use the same visualization libraries, the maps that are published on the web look the same as in desktop GIS.

In the following sections, we will provide a sample configuration to set up a QGIS Server on Linux (Debian, Ubuntu and derivatives) and on Windows. For more information about server plugin development, please read `server_plugins`.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section `gnu_fdl`.

GETTING STARTED

2.1 Installation on Debian-based systems

We will give a short and simple installation how-to for a minimal working configuration on Debian based systems (including Ubuntu and derivatives). However, many other distributions and OSs provide packages for QGIS Server.

Note: In Ubuntu you can use your regular user, prepending `sudo` to commands requiring admin permissions. In Debian you can work as admin (`root`), without using `sudo`.

Requirements and steps to add official QGIS repositories to install QGIS Server on a Debian based system are provided in [QGIS installers page](#). You may want to install at least the latest Long Term Release.

Once the target version repository is configured and QGIS Server installed, you can test the installation with:

```
/usr/lib/cgi-bin/qgis_mapserv.fcgi
```

If you get the following output, the server is correctly installed.

Note: Depending on the version of QGIS, you might see slightly different output reported when you run `qgis_mapserv.fcgi`.

```
QFSFileEngine::open: No file name specified
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment_
↳variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment_
↳variable.
Content-Length: 206
Content-Type: text/xml; charset=utf-8

<ServiceExceptionReport version="1.3.0" xmlns="https://www.opengis.net/ogc">
  <ServiceException code="Service configuration error">Service unknown or_
↳unsupported</ServiceException>
</ServiceExceptionReport>
```

Note: As seen below, QGIS reports a Status 400 code, which correctly identifies the request has failed because there is no active http session. This is not a bug and indicates the server is functioning properly.

```
Application path not initialized
Application path not initialized
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
```

(continues on next page)

(continued from previous page)

```
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment_
↳variable.
"Loading native module /usr/lib/qgis/server/libdummy.so"
"Loading native module /usr/lib/qgis/server/liblandingpage.so"
"Loading native module /usr/lib/qgis/server/libwcs.so"
"Loading native module /usr/lib/qgis/server/libwfs.so"
"Loading native module /usr/lib/qgis/server/libwfs3.so"
"Loading native module /usr/lib/qgis/server/libwms.so"
"Loading native module /usr/lib/qgis/server/libwmts.so"
QFSFileEngine::open: No file name specified
Content-Length: 102
Content-Type: application/json
Server: QGIS FCGI server - QGIS version 3.16.6-Hannover
Status: 400
[{"code": "Bad request error", "description": "Requested URI does not match any_
↳registered API handler"}]
```

Let's add a sample project. You can use your own, or one from [Training demo data](#):

```
mkdir /home/qgis/projects/
cd /home/qgis/projects/
wget https://github.com/qgis/QGIS-Training-Data/archive/release_3.16.zip
unzip release_3.16.zip
mv QGIS-Training-Data-release_3.16/exercise_data/qgis-server-tutorial-data/world.
↳qgs .
mv QGIS-Training-Data-release_3.16/exercise_data/qgis-server-tutorial-data/
↳naturalearth.sqlite .
```

Of course, you can use your favorite GIS software to open this file and take a look at the configuration and available layers.

To properly deploy QGIS server you need a HTTP server. Recommended choices are **Apache** or **NGINX**.

2.1.1 Apache HTTP Server

Note: In the following, please replace `qgis.demo` with the name or IP address of your server.

Install Apache and `mod_fcgid`:

```
apt install apache2 libapache2-mod-fcgid
```

You can run QGIS Server on your default website, or configure a virtualhost specifically for this, as follows.

In the `/etc/apache2/sites-available` directory let's create a file called `qgis.demo.conf`, with this content:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName qgis.demo

    DocumentRoot /var/www/html

    # Apache logs (different than QGIS Server log)
    ErrorLog ${APACHE_LOG_DIR}/qgis.demo.error.log
    CustomLog ${APACHE_LOG_DIR}/qgis.demo.access.log combined

    # Longer timeout for WPS... default = 40
```

(continues on next page)

(continued from previous page)

```

FcgidIOTimeout 120

FcgidInitialEnv LC_ALL "en_US.UTF-8"
FcgidInitialEnv PYTHONIOENCODING UTF-8
FcgidInitialEnv LANG "en_US.UTF-8"

# QGIS log
FcgidInitialEnv QGIS_SERVER_LOG_STDERR 1
FcgidInitialEnv QGIS_SERVER_LOG_LEVEL 0

# default QGIS project
SetEnv QGIS_PROJECT_FILE /home/qgis/projects/world.qgs

# QGIS_AUTH_DB_DIR_PATH must lead to a directory writeable by the Server's FCGI_
↳process user
FcgidInitialEnv QGIS_AUTH_DB_DIR_PATH "/home/qgis/qgisserverdb/"
FcgidInitialEnv QGIS_AUTH_PASSWORD_FILE "/home/qgis/qgisserverdb/qgis-auth.db"

# Set pg access via pg_service file
SetEnv PGSERVICEFILE /home/qgis/.pg_service.conf
FcgidInitialEnv PGPASSFILE "/home/qgis/.pgpass"

# if qgis-server is installed from packages in debian based distros this is_
↳usually /usr/lib/cgi-bin/
# run "locate qgis_mapserv.fcgi" if you don't know where qgis_mapserv.fcgi is
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin/">
    AllowOverride None
    Options +ExecCGI -MultiViews -SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
    Require all granted
</Directory>

<IfModule mod_fcgid.c>
FcgidMaxRequestLen 26214400
FcgidConnectTimeout 60
</IfModule>

</VirtualHost>

```

Further readings:

- [QGIS Server logging](#)
- [pg-service-file](#) in QGIS Server

You can do the above in a linux Desktop system by pasting and saving the above configuration after doing:

```
nano /etc/apache2/sites-available/qgis.demo.conf
```

Note: Some of the configuration options are explained in the Server *environment variables* section.

Let's now create the directories that will store the QGIS Server logs and the authentication database:

```

mkdir -p /var/log/qgis/
chown www-data:www-data /var/log/qgis
mkdir -p /home/qgis/qgisserverdb
chown www-data:www-data /home/qgis/qgisserverdb

```

Note: `www-data` is the Apache user on Debian based systems and we need Apache to have access to those locations or files. The `chown www-data . . .` commands change the owner of the respective directories and files to `www-data`.

We can now enable the [virtual host](#), enable the `fcgid` mod if it's not already enabled:

```
a2enmod fcgid
a2ensite qgis.demo
```

Now restart Apache for the new configuration to be taken into account:

```
systemctl restart apache2
```

Now that Apache knows that he should answer requests to <http://qgis.demo> we also need to setup the client system so that it knows who `qgis.demo` is. We do that by adding `127.0.0.1 qgis.demo` in the `hosts` file. We can do it with `sh -c "echo '127.0.0.1 qgis.demo' >> /etc/hosts"`. Replace `127.0.0.1` with the IP of your server.

Note: Remember that both the `qgis.demo.conf` and `/etc/hosts` files should be configured for your setup to work. You can also test the access to your QGIS Server from other clients on the network (e.g. Windows or macOS machines) by going to their `/etc/hosts` file and point the `myhost` name to whatever IP the server machine has on the network (not `127.0.0.1` as it is the local IP, only accessible from the local machine). On `*nix` machines the `hosts` file is located in `/etc`, while on Windows it's under the `C:\Windows\System32\drivers\etc` directory. Under Windows you need to start your text editor with administrator privileges before opening the `hosts` file.

QGIS Server is now available at <http://qgis.demo>. To check, type in a browser, as in the simple case:

```
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&
↔REQUEST=GetCapabilities
```

2.1.2 NGINX HTTP Server

Note: In the following, please replace `qgis.demo` with the name or IP address of your server.

You can also use QGIS Server with [NGINX](#). Unlike Apache, NGINX does not automatically spawn FastCGI processes. The FastCGI processes are to be started by something else.

Install NGINX:

```
apt install nginx
```

- As a first option, you can use **spawn-fcgi** or **fcgiwrap** to start and manage the QGIS Server processes. Official Debian packages exist for both. When you have no X server running and you need, for example, printing, you can use `xvfb`.
- Another option is to rely on **Systemd**, the init system for GNU/Linux that most Linux distributions use today. One of the advantages of this method is that it requires no other components or processes. It's meant to be simple, yet robust and efficient for production deployments.

NGINX Configuration

The `include fastcgi_params`; used in the previous configuration is important, as it adds the parameters from `/etc/nginx/fastcgi_params`:

```
fastcgi_param QUERY_STRING      $query_string;
fastcgi_param REQUEST_METHOD    $request_method;
fastcgi_param CONTENT_TYPE      $content_type;
fastcgi_param CONTENT_LENGTH    $content_length;

fastcgi_param SCRIPT_NAME       $fastcgi_script_name;
fastcgi_param REQUEST_URI       $request_uri;
fastcgi_param DOCUMENT_URI      $document_uri;
fastcgi_param DOCUMENT_ROOT     $document_root;
fastcgi_param SERVER_PROTOCOL   $server_protocol;
fastcgi_param REQUEST_SCHEME    $scheme;
fastcgi_param HTTPS             $https if_not_empty;

fastcgi_param GATEWAY_INTERFACE CGI/1.1;
fastcgi_param SERVER_SOFTWARE   nginx/$nginx_version;

fastcgi_param REMOTE_ADDR       $remote_addr;
fastcgi_param REMOTE_PORT       $remote_port;
fastcgi_param SERVER_ADDR       $server_addr;
fastcgi_param SERVER_PORT       $server_port;
fastcgi_param SERVER_NAME       $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param REDIRECT_STATUS   200;
```

Moreover, you can use some *Environment variables* to configure QGIS Server. In the NGINX configuration file, `/etc/nginx/nginx.conf`, you have to use `fastcgi_param` instruction to define these variables as shown below:

```
location /qgisserver {
    gzip            off;
    include         fastcgi_params;
    fastcgi_param  QGIS_SERVER_LOG_STDERR 1;
    fastcgi_param  QGIS_SERVER_LOG_LEVEL  0;
    fastcgi_pass   unix:/var/run/qgisserver.socket;
}
```

FastCGI wrappers

Warning: `fcgiwrap` is easier to set up than `spawn-fcgi`, because it's already wrapped in a Systemd service. But it also leads to a solution that is much slower than using `spawn-fcgi`. With `fcgiwrap`, a new QGIS Server process is created on each request, meaning that the QGIS Server initialization process, which includes reading and parsing the QGIS project file, is done on each request. With `spawn-fcgi`, the QGIS Server process remains alive between requests, resulting in much better performance. For that reason, `spawn-fcgi` is recommended for production use.

spawn-fcgi

If you want to use `spawn-fcgi`, the first step is to install the package:

```
apt install spawn-fcgi
```

Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
    gzip            off;
    include         fastcgi_params;
    fastcgi_pass    unix:/var/run/qgisserver.socket;
}
```

And restart NGINX to take into account the new configuration:

```
systemctl restart nginx
```

Finally, considering that there is no default service file for spawn-fcgi, you have to manually start QGIS Server in your terminal:

```
spawn-fcgi -s /var/run/qgisserver.socket \
           -U www-data -G www-data -n \
           /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

QGIS Server is now available at <http://qgis.demo/qgisserver>.

Note: When using spawn-fcgi, you may directly define environment variables before running the server. For example: `export QGIS_SERVER_LOG_STDERR=1`

Of course, you can add an init script to start QGIS Server at boot time or whenever you want. For example with **systemd**, edit the file `/etc/systemd/system/qgis-server.service` with this content:

```
[Unit]
Description=QGIS server
After=network.target

[Service]
;; set env var as needed
;Environment="LANG=en_EN.UTF-8"
;Environment="QGIS_SERVER_PARALLEL_RENDERING=1"
;Environment="QGIS_SERVER_MAX_THREADS=12"
;Environment="QGIS_SERVER_LOG_LEVEL=0"
;Environment="QGIS_SERVER_LOG_STDERR=1"
;; or use a file:
;EnvironmentFile=/etc/qgis-server/env

ExecStart=spawn-fcgi -s /var/run/qgisserver.socket -U www-data -G www-data -n /usr/
↳lib/cgi-bin/qgis_mapserv.fcgi

[Install]
WantedBy=multi-user.target
```

Then enable and start the service:

```
systemctl enable --now qgis-server
```

Warning: With the above commands spawn-fcgi spawns only one QGIS Server process.

fcgiwrap

Using `fcgiwrap` is much easier to setup than `spawn-fcgi` but it's much slower. You first have to install the corresponding package:

```
apt install fcgiwrap
```

Then, introduce the following block in your NGINX server configuration:

```
1 location /qgisserver {
2     gzip            off;
3     include         fastcgi_params;
4     fastcgi_pass    unix:/var/run/fcgiwrap.socket;
5     fastcgi_param   SCRIPT_FILENAME /usr/lib/cgi-bin/qgis_mapserv.fcgi;
6 }
```

Finally, restart NGINX and `fcgiwrap` to take into account the new configuration:

```
systemctl restart nginx
systemctl restart fcgiwrap
```

QGIS Server is now available at <http://qgis.demo.qgisserver>.

Systemd

QGIS Server needs a running X Server to be fully usable, in particular for printing. In the case you already have a running X Server, you can use `systemd` services.

This method, to deploy QGIS Server, relies on two `Systemd` units:

- a `Socket` unit
- and a `Service` unit.

The **QGIS Server Socket** unit defines and creates a file system socket, used by NGINX to start and communicate with QGIS Server. The `Socket` unit has to be configured with `Accept=false`, meaning that the calls to the `accept()` system call are delegated to the process created by the `Service` unit. It is located in `/etc/systemd/system/qgis-server@.socket`, which is actually a template:

```
[Unit]
Description=QGIS Server Listen Socket (instance %i)

[Socket]
Accept=false
ListenStream=/var/run/qgis-server-%i.sock
SocketUser=www-data
SocketGroup=www-data
SocketMode=0600

[Install]
WantedBy=sockets.target
```

Now enable and start sockets:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.socket; done
```

The **QGIS Server Service** unit defines and starts the QGIS Server process. The important part is that the `Service` process' standard input is connected to the socket defined by the `Socket` unit. This has to be configured using `StandardInput=socket` in the `Service` unit configuration located in `/etc/systemd/system/qgis-server@.service`:

```
[Unit]
Description=QGIS Server Service (instance %i)

[Service]
User=www-data
Group=www-data
StandardOutput=null
StandardError=journal
StandardInput=socket
ExecStart=/usr/lib/cgi-bin/qgis_mapserv.fcgi
EnvironmentFile=/etc/qgis-server/env

[Install]
WantedBy=multi-user.target
```

Note: The QGIS Server *environment variables* are defined in a separate file, `/etc/qgis-server/env`. It could look like this:

```
QGIS_PROJECT_FILE=/etc/qgis/myproject.qgs
QGIS_SERVER_LOG_STDERR=1
QGIS_SERVER_LOG_LEVEL=3
```

Now start socket service:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.service; done
```

Finally, for the NGINX HTTP server, lets introduce the configuration for this setup:

```
upstream qgis-server_backend {
    server unix:/var/run/qgis-server-1.sock;
    server unix:/var/run/qgis-server-2.sock;
    server unix:/var/run/qgis-server-3.sock;
    server unix:/var/run/qgis-server-4.sock;
}

server {
    ...

    location /qgis-server {
        gzip off;
        include fastcgi_params;
        fastcgi_pass qgis-server_backend;
    }
}
```

Now restart NGINX for the new configuration to be taken into account:

```
systemctl restart nginx
```

Thanks to Oslandia for sharing their tutorial.

2.1.3 Xvfb

QGIS Server needs a running X Server to be fully usable, in particular for printing. On servers it is usually recommended not to install it, so you may use `xvfb` to have a virtual X environment.

If you're running the Server in graphic/X11 environment then there is no need to install `xvfb`. More info at <https://www.itopen.it/qgis-server-setup-notes/>.

To install the package:

```
apt install xvfb
```

Create the service file, `/etc/systemd/system/xvfb.service`, with this content:

```
[Unit]
Description=X Virtual Frame Buffer Service
After=network.target

[Service]
ExecStart=/usr/bin/Xvfb :99 -screen 0 1024x768x24 -ac +extension GLX +render -
↳noreset

[Install]
WantedBy=multi-user.target
```

Enable, start and check the status of the `xvfb.service`:

```
systemctl enable --now xvfb.service
systemctl status xvfb.service
```

Then, according to your HTTP server, you should configure the **DISPLAY** parameter or directly use **xvfb-run**.

With Apache

Then you can configure the **DISPLAY** parameter.

With Apache you just add to your *FastCGI* configuration (see above):

```
FcgidInitialEnv DISPLAY      ":99"
```

Now restart Apache for the new configuration to be taken into account:

```
systemctl restart apache2
```

With NGINX

Then you can directly use **xvfb-run** or configure the **DISPLAY** parameter.

- With spawn-fcgi using `xvfb-run`:

```
xvfb-run /usr/bin/spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi \
-s /tmp/qgisserver.socket \
-G www-data -U www-data -n
```

- With the **DISPLAY** environment variable in the HTTP server configuration.

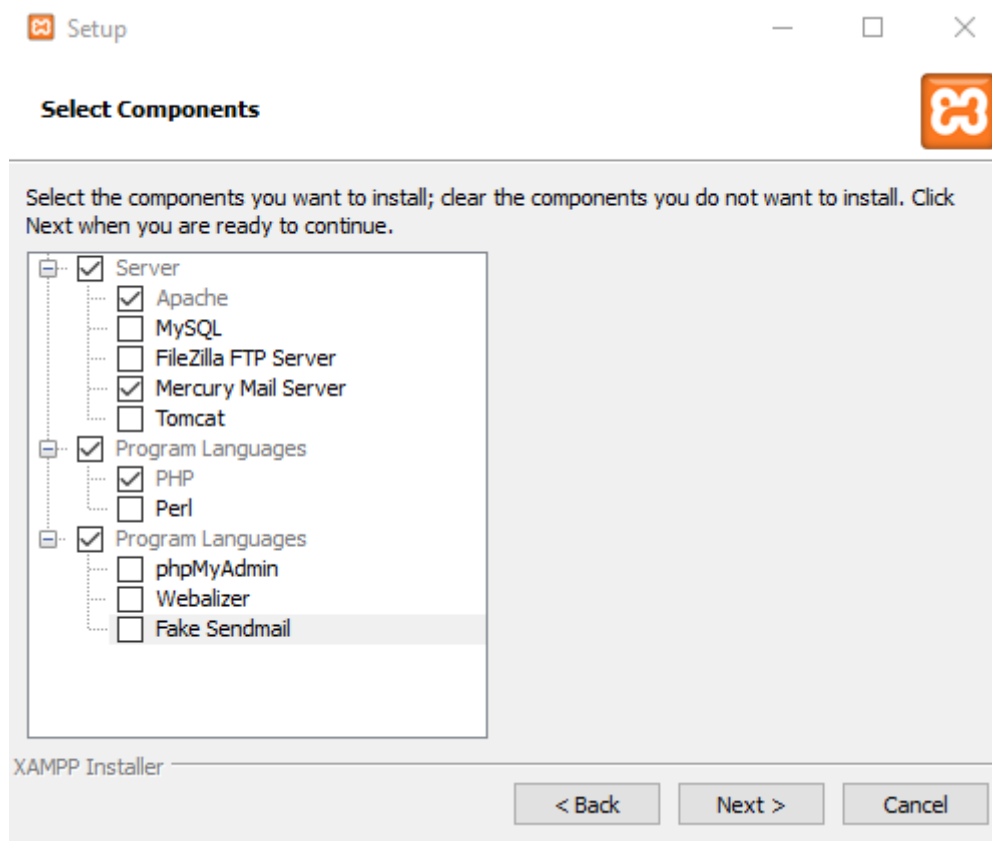
```
fastcgi_param DISPLAY      ":99";
```

2.2 Installation on Windows

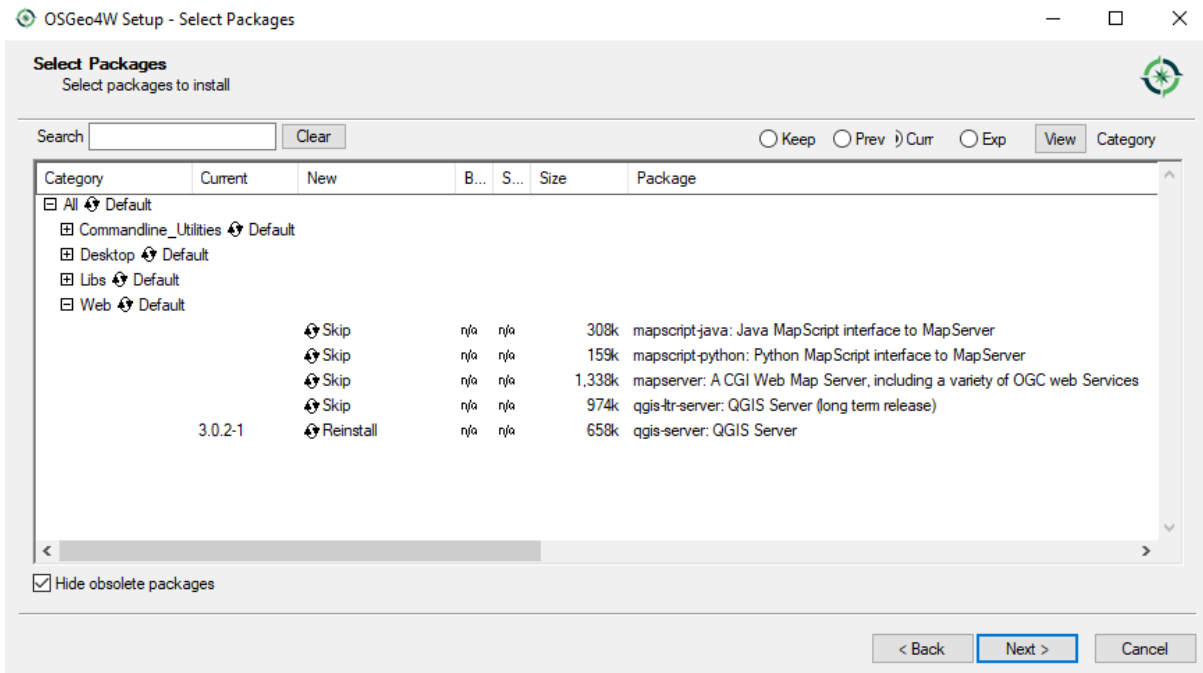
QGIS Server can also be installed on Windows systems. While the QGIS Server package is available in the 64 bit version of the OSGeo4W network installer (<https://qgis.org/en/site/forusers/download.html>) there is no Apache (or other web server) package available, so this must be installed by other means.

A simple procedure is the following:

1. Download the XAMPP installer (<https://www.apachefriends.org/download.html>) for Windows and install Apache



2. Download the OSGeo4W installer, follow the “Advanced Install” and install both the QGIS Desktop and QGIS Server packages



3. Edit the httpd.conf file (C:\xampp\apache\conf\httpd.conf if the default installation paths have been used) and make the following changes:

From:

```
ScriptAlias /cgi-bin/ "C:/xampp/cgi-bin/"
```

To:

```
ScriptAlias /cgi-bin/ "C:/OSGeo4W64/apps/qgis/bin/"
```

From:

```
<Directory "C:/xampp/cgi-bin">
  AllowOverride None
  Options None
  Require all granted
</Directory>
```

To:

```
<Directory "C:/OSGeo4W64/apps/qgis/bin">
  SetHandler cgi-script
  AllowOverride None
  Options ExecCGI
  Order allow,deny
  Allow from all
  Require all granted
</Directory>
```

From:

```
AddHandler cgi-script .cgi .pl .asp
```

To:

```
AddHandler cgi-script .cgi .pl .asp .exe
```

4. Then at the bottom of httpd.conf add:

```
SetEnv GDAL_DATA "C:\OSGeo4W64\share\gdal"  
SetEnv QGIS_AUTH_DB_DIR_PATH "C:\OSGeo4W64\apps\qgis\resources"  
SetEnv PYTHONHOME "C:\OSGeo4W64\apps\Python37"  
SetEnv PATH "C:\OSGeo4W64\bin;C:\OSGeo4W64\apps\qgis\bin;C:\OSGeo4W64\apps\Qt5\  
↪bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem"  
SetEnv QGIS_PREFIX_PATH "C:\OSGeo4W64\apps\qgis"  
SetEnv QT_PLUGIN_PATH "C:\OSGeo4W64\apps\qgis\qtplugins;C:\OSGeo4W64\apps\Qt5\  
↪plugins"
```

5. Restart the Apache web server from the XAMPP Control Panel and open browser window to testing a GetCapabilities request to QGIS Server

```
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi.exe?SERVICE=WMS&VERSION=1.3.0&  
↪REQUEST=GetCapabilities
```

2.3 Serve a project

Now that QGIS Server is installed and running, we just have to use it.

Obviously, we need a QGIS project to work on. Of course, you can fully customize your project by defining contact information, precise some restrictions on CRS or even exclude some layers. Everything you need to know about that is described later in *Configure your project*.

But for now, we are going to use a simple project already configured and previously downloaded in `/home/qgis/projects/world.qgs`, as described above.

By opening the project and taking a quick look on layers, we know that 4 layers are currently available:

- airports
- places
- countries
- countries_shapeburst

You don't have to understand the full request for now but you may retrieve a map with some of the previous layers thanks to QGIS Server by doing something like this in your web browser to retrieve the *countries* layer:

```
http://qgis.demo/qgisserver?  
MAP=/home/qgis/projects/world.qgs&  
LAYERS=countries&  
SERVICE=WMS&  
VERSION=1.3.0&  
REQUEST=GetMap&  
CRS=EPSG:4326&  
WIDTH=400&  
HEIGHT=200&  
BBOX=-90,-180,90,180
```

If you obtain the next image, then QGIS Server is running correctly:

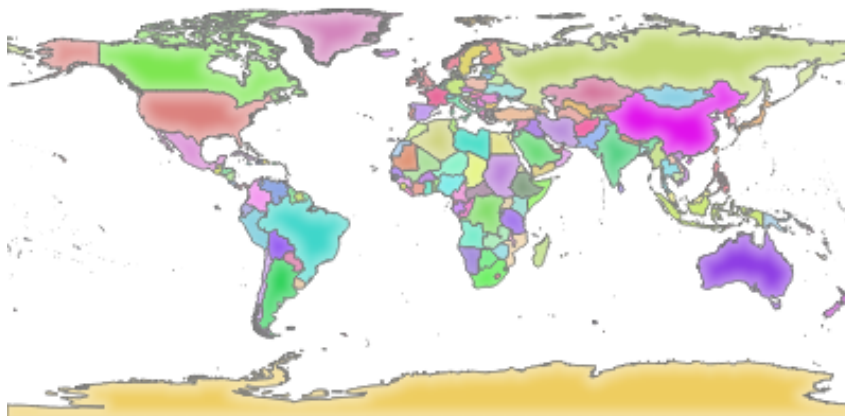


Fig. 2.1: Server response to a basic GetMap request

Note that you may define **QGIS_PROJECT_FILE** environment variable to use a project by default instead of giving a **MAP** parameter (see *Environment variables*).

For example with spawn-fcgi:

```
export QGIS_PROJECT_FILE=/home/qgis/projects/world.qgs
spawn-fcgi -f /usr/lib/bin/cgi-bin/qgis_mapserv.fcgi \
-s /var/run/qgisserver.socket \
-U www-data -G www-data -n
```

2.4 Configure your project

To provide a new QGIS Server WMS, WFS or WCS, you have to create a QGIS project file with some data or use one of your current project. Define the colors and styles of the layers in QGIS and the project CRS, if not already defined.

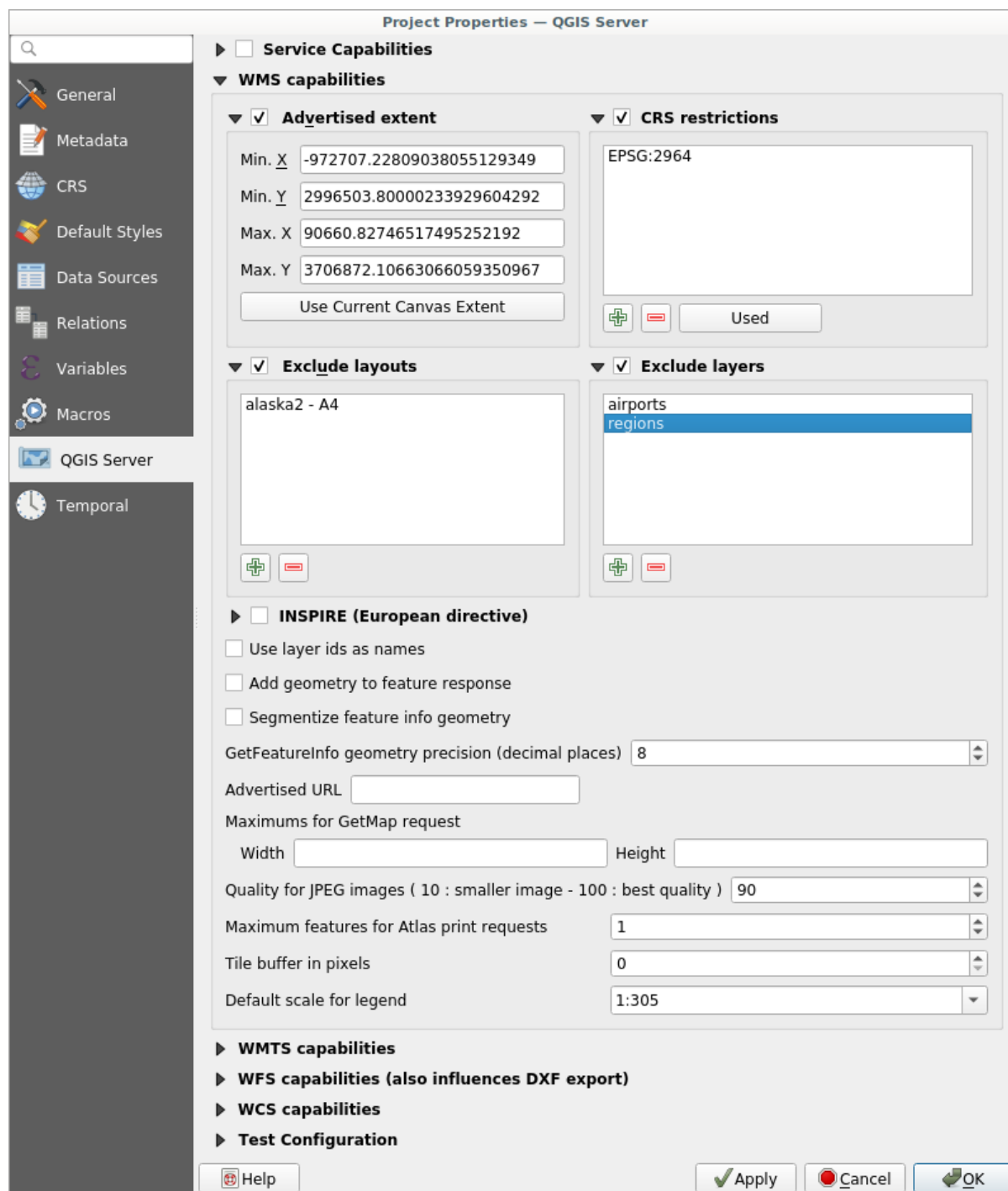





Fig. 2.2: Definitions for a QGIS Server WMS/WFS/WCS project

Then, go to the *QGIS Server* menu of the *Project ► Properties...* dialog and provide some information about the OWS in the fields under *Service Capabilities*. This will appear in the *GetCapabilities* response of the WMS, WFS or WCS. If you don't check *Service capabilities*, QGIS Server will use the information given in the `wms_metadata.xml` file located in the `cgi-bin` folder.

2.4.1 WMS capabilities

In the *WMS capabilities* section, you can define the extent advertised in the WMS GetCapabilities response by entering the minimum and maximum X and Y values in the fields under *Advertised extent*. Clicking *Use Current Canvas Extent* sets these values to the extent currently displayed in the QGIS map canvas. By checking *CRS restrictions*, you can restrict in which coordinate reference systems (CRS) QGIS Server will offer to render maps. It is recommended that you restrict the offered CRS as this reduces the size of the WMS GetCapabilities response. Use the  button below to select those CRSs from the Coordinate Reference System Selector, or click *Used* to add the CRSs used in the QGIS project to the list.

If you have print layouts defined in your project, they will be listed in the *GetProjectSettings* response, and they can be used by the GetPrint request to create prints, using one of the print layouts as a template. This is a QGIS-specific extension to the WMS 1.3.0 specification. If you want to exclude any print layout from being published by the WMS, check *Exclude layouts* and click the  button below. Then, select a print layout from the *Select print layout* dialog in order to add it to the excluded layouts list.

If you want to exclude any layer or layer group from being published by the WMS, check *Exclude Layers* and click the  button below. This opens the *Select restricted layers and groups* dialog, which allows you to choose the layers and groups that you don't want to be published. Use the *Shift* or *Ctrl* key if you want to select multiple entries. It is recommended that you exclude from publishing the layers that you don't need as this reduces the size of the WMS GetCapabilities response which leads to faster loading times on the client side.

If you check *Use layer ids as name*, layer ids will be used to reference layers in the *GetCapabilities* response or *GetMap* `LAYERS` parameter. If not, layer name or short name if defined (see `vectorservermenu`) is used.

You can receive requested *GetFeatureInfo* as plain text, XML and GML. The default is XML.

If you wish, you can check *Add geometry to feature response*. This will include the bounding box for each feature in the *GetFeatureInfo* response. See also the `WITH_GEOMETRY` parameter.

As many web clients can't display circular arcs in geometries you have the option to segmentize the geometry before sending it to the client in a *GetFeatureInfo* response. This allows such clients to still display a feature's geometry (e.g. for highlighting the feature). You need to check the *Segmentize feature info geometry* to activate the option.

You can also use the *GetFeatureInfo geometry precision* option to set the precision of the *GetFeatureInfo* geometry. This enables you to save bandwidth when you don't need the full precision.

If you want QGIS Server to advertise specific request URLs in the WMS GetCapabilities response, enter the corresponding URL in the *Advertised URL* field.

Furthermore, you can restrict the maximum size of the maps returned by the *GetMap* request by entering the maximum width and height into the respective fields under *Maximums for GetMap request*.

You can change the *Quality for JPEG images* factor. The quality factor must be in the range 0 to 100. Specify 0 for maximum compression, 100 for no compression.

You can change the limit for atlas features to be printed in one request by setting the *Maximum features for Atlas print requests* field.

When QGIS Server is used in tiled mode (see *TILED parameter*), you can set the *Tile buffer in pixels*. The recommended value is the size of the largest symbol or line width in your QGIS project.

If one of your layers uses the Map Tip display (i.e. to show text using expressions) this will be listed inside the *GetFeatureInfo* output. If the layer uses a Value Map for one of its attributes, this information will also be shown in the *GetFeatureInfo* output.

2.4.2 WFS capabilities

In the *WFS capabilities* area you can select the layers you want to publish as WFS, and specify if they will allow update, insert and delete operations. If you enter a URL in the *Advertised URL* field of the *WFS capabilities* section, QGIS Server will advertise this specific URL in the WFS GetCapabilities response.

2.4.3 WCS capabilities

In the *WCS capabilities* area, you can select the layers that you want to publish as WCS. If you enter a URL in the *Advertised URL* field of the *WCS capabilities* section, QGIS Server will advertise this specific URL in the WCS GetCapabilities response.

2.4.4 Fine tuning your OWS

For vector layers, the *Fields* menu of the *Layer ► Layer Properties* dialog allows you to define for each attribute if it will be published or not. By default, all the attributes are published by your WMS and WFS. If you don't want a specific attribute to be published, uncheck the corresponding checkbox in the *WMS* or *WFS* column.

You can overlay watermarks over the maps produced by your WMS by adding text annotations or SVG annotations to the project file. See the *sec_annotations* section for instructions on creating annotations. For annotations to be displayed as watermarks on the WMS output, the *Fixed map position* checkbox in the *Annotation text* dialog must be unchecked. This can be accessed by double clicking the annotation while one of the annotation tools is active. For SVG annotations, you will need either to set the project to save absolute paths (in the *General* menu of the *Project ► Properties...* dialog) or to manually modify the path to the SVG image so that it represents a valid relative path.

2.5 Integration with third parties

QGIS Server provides standard OGC web services like [WMS](#), [WFS](#), etc. thus it can be used by a wide variety of end user tools.

2.5.1 Integration with QGIS Desktop

QGIS Desktop is the map designer where QGIS Server is the map server. The maps or QGIS projects will be served by the QGIS Server to provide OGC standards. These QGIS projects can either be files or entries in a database (by using *Project ► Save to ► PostgreSQL* in QGIS Desktop).

Furthermore, dedicated update workflow must be established to refresh a project used by a QGIS Server (ie. copy project files into server location and restart QGIS Server). For now, automated processes (as server reloading over message queue service) are not implemented yet.

2.5.2 Integration with MapProxy

[MapProxy](#) is a tile cache server and as it can read and serve any WMS/WMTS map server, it can be directly connected to QGIS server web services and improve end user experience.

2.5.3 Integration with QWC2

QWC2 is a responsive web application dedicated to QGIS Server. It helps you to build a highly customized map viewer with layer selection, feature info, etc.. Also many plugins are available like authentication or print service, the full list is available is this [repository](#).

SERVICES

QGIS Server is able to serve data according to standard protocols as described by the **Open Geospatial Consortium (OGC)**:

- WMS 1.1.1 and 1.3.0
- WFS 1.0.0 and 1.1.0
- OGC API - Features (WFS3)
- WCS 1.1.1
- WMTS 1.0.0

Extra vendor parameters and requests are supported in addition to the original standard that greatly enhance the possibilities of customizing its behavior thanks to the QGIS rendering engine.

3.1 Web Map Service (WMS)

The **1.1.1** and **1.3.0** WMS standards implemented in QGIS Server provide a HTTP interface to request map or legend images generated from a QGIS project. A typical WMS request defines the QGIS project to use, the layers to render as well as the image format to generate. Basic support is also available for **Styled Layer Descriptor (SLD)**.

Specifications:

- [WMS 1.1.1](#)
- [WMS 1.3.0](#)
- [SLD 1.1.0 WMS profile](#)

Standard requests provided by QGIS Server:

Request	Description
GetCapabilities	Returns XML metadata with information about the server
<i>GetMap</i>	Returns a map
<i>GetFeatureInfo</i>	Retrieves data (geometry and values) for a pixel location
<i>GetLegendGraphics</i>	Returns legend symbols

Vendor requests provided by QGIS Server:

Request	Description
<i>GetPrint</i>	Returns a QGIS composition
<i>GetProjectSettings</i>	Returns specific information about QGIS Server

3.1.1 GetMap

Standard parameters for the **GetMap** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parameter	Required	Description
<i>SERVICE</i>	Yes	Name of the service (WMS)
<i>VERSION</i>	Yes	Version of the service
<i>REQUEST</i>	Yes	Name of the request (GetMap)
<i>LAYERS</i>	No	Layers to display
<i>STYLES</i>	No	Layers' style
<i>SRS / CRS</i>	Yes	Coordinate reference system
<i>BBOX</i>	Yes	Map extent
<i>WIDTH</i>	Yes	Width of the image in pixels
<i>HEIGHT</i>	Yes	Height of the image in pixels
<i>FORMAT</i>	No	Image format
<i>TRANSPARENT</i>	No	Transparent background
<i>SLD</i>	No	URL of an SLD to be used for styling
<i>SLD_BODY</i>	No	In-line SLD (XML) to be used for styling

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
<i>MAP</i>	Yes	Specify the QGIS project file
<i>BGCOLOR</i>	No	Specify the background color
<i>DPI</i>	No	Specify the output resolution
<i>IMAGE_QUALITY</i>	No	JPEG compression
<i>OPACITIES</i>	No	Opacity for layer or group
<i>FILTER</i>	No	Subset of features
<i>SELECTION</i>	No	Highlight features
<i>FILE_NAME</i>	No	File name of the downloaded file Only for <i>FORMAT=application/dxf</i>
<i>FORMAT_OPTIONS</i>	No	Options of the specified file format Only for <i>FORMAT=application/dxf</i>
<i>TILED</i>	No	Working in <i>tiled mode</i>

URL example:

```
http://localhost/qgis_server?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&STYLES=style1,default,style3
&OPACITIES=125,200,125
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400
&FORMAT=image/png
&TRANSPARENT=TRUE
&DPI=300
&TILED=TRUE
```

SERVICE

This parameter has to be WMS.

For example:

```
http://localhost/qgisserver?
SERVICE=WMS
&...
```

VERSION

This parameter allows to specify the version of the service to use. Available values for the VERSION parameter are:

- 1.1.1
- 1.3.0

According to the version number, slight differences have to be expected as explained later for the next parameters:

- CRS / SRS
- BBOX

REQUEST

This parameter is GetMap in case of the **GetMap** request.

LAYERS

This parameter allows to specify the layers to display on the map. Names have to be separated by a comma.

In addition, QGIS Server introduced some options to select layers by:

- a short name
- the layer id

The short name of a layer may be configured through *Properties* ► *Metadata* in layer menu. If the short name is defined, then it's used by default instead of the layer's name:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mynickname1,mynickname2
&...
```

Moreover, there's a project option allowing to select layers by their id in *OWS Server* ► *WMS capabilities* menu of the *Project* ► *Properties...* dialog. To activate this option, the checkbox *Use layer ids as names* has to be selected.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayerid1,mylayerid2
&...
```

STYLES

This parameter can be used to specify a layer's style for the rendering step. Styles have to be separated by a comma. The name of the default style is `default`.

SRS / CRS

This parameter allows to indicate the map output Spatial Reference System in WMS **1.1.1** and has to be formed like `EPSG:XXXX`. Note that `CRS` is also supported if current version is **1.1.1**.

For WMS **1.3.0**, `CRS` parameter is preferable but `SRS` is also supported.

Note that if both `CRS` and `SRS` parameters are indicated in the request, then it's the current version indicated in `VERSION` parameter which is decisive.

In the next case, the `SRS` parameter is kept whatever the `VERSION` parameter because `CRS` is not indicated:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&REQUEST=GetMap  
&VERSION=1.3.0  
&SRS=EPSG:2854  
&...
```

In the next case, the `SRS` parameter is kept instead of `CRS` because of the `VERSION` parameter:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&REQUEST=GetMap  
&VERSION=1.1.1  
&CRS=EPSG:4326  
&SRS=EPSG:2854  
&...
```

In the next case, the `CRS` parameter is kept instead of `SRS` because of the `VERSION` parameter:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&REQUEST=GetMap  
&VERSION=1.3.0  
&CRS=EPSG:4326  
&SRS=EPSG:2854  
&...
```

BBOX

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

The `BBOX` parameter is formed like `min_a,min_b,max_a,max_b` but `a` and `b` axis definition is different according to the current `VERSION` parameter:

- in WMS **1.1.1**, the axis ordering is always east/north
- in WMS **1.3.0**, the axis ordering depends on the CRS authority

For example in case of `EPSG:4326` and WMS **1.1.1**, `a` is the longitude (east) and `b` the latitude (north), leading to a request like:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.1
&SRS=epsg:4326
&BBOX=-180,-90,180,90
&...
```

But in case of WMS **1.3.0**, the axis ordering defined in the EPSG database is north/east so a is the latitude and b the longitude:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=epsg:4326
&BBOX=-90,-180,90,180
&...
```

WIDTH

This parameter allows to specify the width in pixels of the output image.

HEIGHT

This parameter allows to specify the height in pixels of the output image.

FORMAT

This parameter may be used to specify the format of map image. Available values are:

- jpg
- jpeg
- image/jpeg
- image/png
- image/png; mode=1bit
- image/png; mode=8bit
- image/png; mode=16bit
- application/dxf Only layers that have read access in the WFS service are exported in the DXF format.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS&VERSION=1.3.0
&REQUEST=GetMap
&FORMAT=application/dxf
&LAYERS=Haltungen,Normschacht,Spezialbauwerke
&STYLES=
&CRS=EPSG%3A21781&BBOX=696136.28844801,245797.12108743,696318.91114315,245939.
↪25832905
&WIDTH=1042
&HEIGHT=811
&FORMAT_OPTIONS=MODE:SYMBOLLAYERSYMBOLOLOGY;SCALE:250&FILE_NAME=plan.dxf
```

TRANSPARENT

This boolean parameter can be used to specify the background transparency. Available values are (not case sensitive):

- TRUE
- FALSE

However, this parameter is ignored if the format of the map image indicated with `FORMAT` is different from PNG.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in *GetMap parameters table*, `MAP` is mandatory because a request needs a QGIS project to actually work. However, the `QGIS_PROJECT_FILE` environment variable may be used to define a default QGIS project. In this specific case, `MAP` is not longer a required parameter. For further information you may refer to *Advanced configuration*.

BGCOLOR

This parameter allows to indicate a background color for the map image. However it cannot be combined with `TRANSPARENT` parameter in case of PNG images (transparency takes priority). The colour may be literal or in hexadecimal notation.

URL example with the literal notation:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&REQUEST=GetMap  
&VERSION=1.3.0  
&BGCOLOR=green  
&...
```

URL example with the hexadecimal notation:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&REQUEST=GetMap  
&VERSION=1.3.0  
&BGCOLOR=0x00FF00  
&...
```

DPI

This parameter can be used to specify the requested output resolution.

IMAGE_QUALITY

This parameter is only used for JPEG images. By default, the JPEG compression is `-1`.

You can change the default per QGIS project in the *OWS Server ► WMS capabilities* menu of the *Project ► Properties...* dialog. If you want to override it in a `GetMap` request you can do it using the `IMAGE_QUALITY` parameter.

OPACITIES

Comma separated list of opacity values. Opacity can be set on layer or group level. Allowed values range from 0 (fully transparent) to 255 (fully opaque).

FILTER

A subset of layers can be selected with the `FILTER` parameter. The syntax is basically the same as for the QGIS subset string. However, there are some restrictions to avoid SQL injections into databases via QGIS Server. If a dangerous string is found in the parameter, QGIS Server will return the next error:

```
<ServiceExceptionReport>
  <ServiceException code="Security">The filter string XXXXXXXXX has been rejected.
  ↳because of security reasons.
  Note: Text strings have to be enclosed in single or double quotes. A space.
  ↳between each word / special character is mandatory.
  Allowed Keywords and special characters are IS, NOT, NULL, AND, OR, IN, =, <, =<, >, >=, !=,
  ↳', ', (, ), DMETAPHONE, SOUNDEX.
  Not allowed are semicolons in the filter expression.</ServiceException>
</ServiceExceptionReport>
```

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2,mylayer3
&FILTER=mylayer1:"col1";mylayer1,mylayer2:"col2" = 'blabla'
&...
```

In this example, the same filter (field `col2` equals the string `blabla`) is applied to layers `mylayer1` and `mylayer2`, while the filter on `col1` is only applied to `mylayer1`.

Note: It is possible to make attribute searches via `GetFeatureInfo` and omit the `X/Y` parameter if a `FILTER` is there. QGIS Server then returns info about the matching features and generates a combined bounding box in the XML output.

SELECTION

The `SELECTION` parameter can highlight features from one or more layers. Vector features can be selected by passing comma separated lists with feature ids.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2
&SELECTION=mylayer1:3,6,9;mylayer2:1,5,6
&...
```

The following image presents the response from a `GetMap` request using the `SELECTION` option e.g. `http://myserver.com/...&SELECTION=countries:171,65`.

As those features id's correspond in the source dataset to **France** and **Romania** they're highlighted in yellow.



Fig. 3.1: Server response to a GetMap request with SELECTION parameter

FORMAT-OPTIONS

This parameter can be used to specify options for the selected format. Only for `FORMAT=application/dxf`. A list of key:value pairs separated by semicolon:

- **SCALE**: to be used for symbology rules, filters and styles (not actual scaling of the data - data remains in the original scale).
- **MODE**: corresponds to the export options offered in the QGIS Desktop DXF export dialog. Possible values are `NOSYMBOLLOGY`, `FEATURESYMBOLLOGY` and `SYMBOLLAYERSYMBOLLOGY`.
- **LAYERSATTRIBUTES**: specify a field that contains values for DXF layer names - if not specified, the original QGIS layer names are used.
- **USE_TITLE_AS_LAYERNAME**: if enabled, the title of the layer will be used as layer name.
- **CODEC**: specify a codec to be used for encoding. Default is `ISO-8859-1` check the QGIS desktop DXF export dialog for valid values.
- **NO_MTEXT**: Use `TEXT` instead of `MTEXT` for labels.
- **FORCE_2D**: Force 2D output. This is required for polyline width.

TILED

For performance reasons, QGIS Server can be used in tiled mode. In this mode, the client requests several small fixed size tiles, and assembles them to form the whole map. Doing this, symbols at or near the boundary between two tiles may appear cut, because they are only present in one of the tile.

Set the `TILED` parameter to `TRUE` to tell QGIS Server to work in *tiled* mode, and to apply the *Tile buffer* configured in the QGIS project (see *Configure your project*).

When `TILED` is `TRUE` and when a non-zero Tile buffer is configured in the QGIS project, features outside the tile extent are drawn to avoid cut symbols at tile boundaries.

`TILED` defaults to `FALSE`.

3.1.2 GetFeatureInfo

Standard parameters for the **GetFeatureInfo** request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

Parameter	Required	Description
<i>SERVICE</i>	Yes	Name of the service (WMS)
<i>VERSION</i>	No	Version of the service
<i>REQUEST</i>	Yes	Name of the request (GetFeatureInfo)
<i>QUERY_LAYERS</i>	Yes	Layers to query
<i>LAYERS</i>	Yes	Layers to display (identical to <i>QUERY_LAYERS</i>)
<i>STYLES</i>	No	Layers' style
<i>SRS</i> / <i>CRS</i>	Yes	Coordinate reference system
<i>BBOX</i>	No	Map extent
<i>WIDTH</i>	Yes	Width of the image in pixels
<i>HEIGHT</i>	Yes	Height of the image in pixels
<i>TRANSPARENT</i>	No	Transparent background
<i>INFO_FORMAT</i>	No	Output format
<i>FEATURE_COUNT</i>	No	Maximum number of features to return
<i>I</i>	No	Pixel column of the point to query
<i>X</i>	No	Same as <i>I</i> parameter, but in WMS 1.1.1
<i>J</i>	No	Pixel row of the point to query
<i>Y</i>	No	Same as <i>J</i> parameter, but in WMS 1.1.1
<i>WMS_PRECISION</i>	No	The precision (number of digits) to be used when returning geometry (see <i>how to add geometry to feature response</i>). The default value is <code>-1</code> meaning that the precision defined in the project is used.

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
<i>MAP</i>	Yes	Specify the QGIS project file
<i>FILTER</i>	No	Subset of features
<i>FI_POINT_TOLERANCE</i>	No	Tolerance in pixels for point layers
<i>FI_LINE_TOLERANCE</i>	No	Tolerance in pixels for line layers
<i>FI_POLYGON_TOLERANCE</i>	No	Tolerance in pixels for polygon layers
<i>FILTER_GEOM</i>	No	Geometry filtering
<i>WITH_MAPTIP</i>	No	Add map tips to the output
<i>WITH_GEOMETRY</i>	No	Add geometry to the output

URL example:

```
http://localhost/qgisserver?  
SERVICE=WMS  
&VERSION=1.3.0  
&REQUEST=GetMap  
&MAP=/home/qgis/projects/world.qgs  
&LAYERS=mylayer1,mylayer2,mylayer3  
&CRS=EPSG:4326  
&WIDTH=400  
&HEIGHT=400  
&INFO_FORMAT=text/xml  
&TRANSPARENT=TRUE  
&QUERY_LAYERS=mylayer1  
&FEATURE_COUNT=3  
&I=250  
&J=250
```

REQUEST

This parameter is `GetFeatureInfo` in case of the **GetFeatureInfo** request.

INFO_FORMAT

This parameter may be used to specify the format of the result. Available values are:

- `text/xml`
- `text/html`
- `text/plain`
- `application/vnd.ogc.gml`
- `application/json`

QUERY_LAYERS

This parameter specifies the layers to display on the map. Names are separated by a comma.

In addition, QGIS Server introduces options to select layers by:

- short name
- layer id

See the `LAYERS` parameter defined in *GetMap* for more information.

FEATURE_COUNT

This parameter specifies the maximum number of features per layer to return. For example if `QUERY_LAYERS` is set to `layer1,layer2` and `FEATURE_COUNT` is set to 3 then a maximum of 3 features from `layer1` will be returned. Likewise a maximum of 3 features from `layer2` will be returned.

By default, only 1 feature per layer is returned.

I

This parameter, defined in WMS 1.3.0, allows you to specify the pixel column of the query point.

X

Same parameter as I, but defined in WMS 1.1.1.

J

This parameter, defined in WMS 1.3.0, allows you to specify the pixel row of the query point.

Y

Same parameter as J, but defined in WMS 1.1.1.

FI_POINT_TOLERANCE

This parameter specifies the tolerance in pixels for point layers.

FI_LINE_TOLERANCE

This parameter specifies the tolerance in pixels for line layers.

FI_POLYGON_TOLERANCE

This parameter specifies the tolerance in pixels for polygon layers.

FILTER_GEOM

This parameter specifies a WKT geometry with which features have to intersect.

WITH_MAPTIP

This parameter specifies whether to add map tips to the output.

Available values are (not case sensitive):

- TRUE
- FALSE

WITH_GEOMETRY

This parameter specifies whether to add geometries to the output. To use this feature you must first enable the *Add geometry to feature response* option in the QGIS project. See *Configure your project*.

Available values are (not case sensitive):

- TRUE
- FALSE

3.1.3 GetPrint

QGIS Server has the capability to create print layout output in pdf or pixel format. Print layout windows in the published project are used as templates. In the **GetPrint** request, the client has the possibility to specify parameters of the contained layout maps and labels.

Parameters for the **GetPrint** request:

Parameter	Required	Description
<i>MAP</i>	Yes	Specify the QGIS project file
<i>SERVICE</i>	Yes	Name of the service (WMS)
<i>VERSION</i>	No	Version of the service
<i>REQUEST</i>	Yes	Name of the request (GetPrint)
<i>LAYERS</i>	No	Layers to display
<i>TEMPLATE</i>	Yes	Layout template to use
<i>SRS / CRS</i>	Yes	Coordinate reference system
<i>FORMAT</i>	No	Output format
<i>ATLAS_PK</i>	No	Atlas features
<i>STYLES</i>	No	Layers' style
<i>TRANSPARENT</i>	No	Transparent background
<i>OPACITIES</i>	No	Opacity for layer or group
<i>SELECTION</i>	No	Highlight features
<i>mapX:EXTENT</i>	No	Extent of the map 'X'
<i>mapX:LAYERS</i>	No	Layers of the map 'X'
<i>mapX:STYLES</i>	No	Layers' style of the map 'X'
<i>mapX:SCALE</i>	No	Layers' scale of the map 'X'
<i>mapX:ROTATION</i>	No	Rotation of the map 'X'
<i>mapX:GRID_INTERVAL_X</i>	No	Grid interval on x axis of the map 'X'
<i>mapX:GRID_INTERVAL_Y</i>	No	Grid interval on y axis of the map 'X'

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetPrint
&MAP=/home/qgis/projects/world.qgs
&CRS=EPSG:4326
&FORMAT=png
&TEMPLATE=Layout%201
&map0:EXTENT=-180,-90,180,90
&map0:LAYERS=mylayer1,mylayer2,mylayer3
&map0:OPACITIES=125,200,125
&map0:ROTATION=45
```

Note that the layout template may contain more than one map. In this way, if you want to configure a specific map, you have to use *mapX*: parameters where X is a positive number that you can retrieve thanks to the **GetProjectSettings** request.

For example:

```
<WMS_Capabilities>
...
<ComposerTemplates xsi:type="wms:_ExtendedCapabilities">
<ComposerTemplate width="297" height="210" name="Druckzusammenstellung 1">
<ComposerMap width="171" height="133" name="map0"/>
<ComposerMap width="49" height="46" name="map1"/></ComposerTemplate>
</ComposerTemplates>
...
</WMS_Capabilities>
```

REQUEST

This parameter has to be `GetPrint` for the **GetPrint** request.

TEMPLATE

This parameter can be used to specify the name of a layout template to use for printing.

FORMAT

This parameter specifies the format of map image. Available values are:

- `png` (default value)
- `image/png`
- `jpg`
- `jpeg`
- `image/jpeg`
- `svg`
- `image/svg`
- `image/svg+xml`
- `pdf`
- `application/pdf`

If the `FORMAT` parameter is different from one of these values, then an exception is returned.

ATLAS_PK

This parameter allows activation of Atlas rendering by indicating which features we want to print. In order to retrieve an atlas with all features, the `*` symbol may be used (according to the maximum number of features allowed in the project configuration).

When `FORMAT` is `pdf`, a single PDF document combining the feature pages is returned. For all other formats, a single page is returned.

mapX:EXTENT

This parameter specifies the extent for a layout map item as `xmin,ymin,xmax,ymax`.

mapX:ROTATION

This parameter specifies the map rotation in degrees.

mapX:GRID_INTERVAL_X

This parameter specifies the grid line density in the X direction.

mapX:GRID_INTERVAL_Y

This parameter specifies the grid line density in the Y direction.

mapX:SCALE

This parameter specifies the map scale for a layout map item. This is useful to ensure scale based visibility of layers and labels even if client and server may have different algorithms to calculate the scale denominator.

mapX:LAYERS

This parameter specifies the layers for a layout map item. See *GetMap Layers* for more information on this parameter.

mapX:STYLES

This parameter specifies the layers' styles defined in a specific layout map item. See *GetMap Styles* for more information on this parameter.

3.1.4 GetLegendGraphics

Several additional parameters are available to change the size of the legend elements:

- **BOXSPACE** space between legend frame and content (mm)
- **FORMAT**, `image/jpeg`, `image/png` or `application/json`. For JSON, symbols are encoded with Base64 and most other options related to layout or fonts are not taken into account because the legend must be built on the client side.
- **LAYERSPACE** vertical space between layers (mm)
- **LAYERTITLESPACE** vertical space between layer title and items following (mm)
- **SYMBOLSPACE** vertical space between symbol and item following (mm)
- **ICONLABELSPACE** horizontal space between symbol and label text (mm)
- **SYMBOLWIDTH** width of the symbol preview (mm)
- **SYMBOLHEIGHT** height of the symbol preview (mm)

These parameters change the font properties for layer titles and item labels:

- **LAYERFONTFAMILY / ITEMFONTFAMILY** font family for layer title / item text
- **LAYERFONTBOLD / ITEMFONTBOLD** TRUE to use a bold font
- **LAYERFONTSIZE / ITEMFONTSIZE** Font size in point
- **LAYERFONTITALIC / ITEMFONTITALIC** TRUE to use italic font
- **LAYERFONTCOLOR / ITEMFONTCOLOR** Hex color code (e.g. #FF0000 for red)
- **LAYERTITLE** FALSE to get only the legend graphics without layer title
- **RULELABEL**:
 - FALSE legend graphics without item labels
 - AUTO hide item label for layers with *Single symbol* rendering

Content based legend. These parameters let the client request a legend showing only the symbols for the features falling into the requested area:

- **BBOX** the geographical area for which the legend should be built
- **CRS / SRS** the coordinate reference system adopted to define the BBOX coordinates
- **SRCWIDTH / SRCHEIGHT** if set these should match the WIDTH and HEIGHT parameters of the GetMap request, to let QGIS Server scale symbols according to the map view image size.

Content based legend features are based on the [UMN MapServer implementation](#):

- **SHOWFEATURECOUNT** if set to TRUE adds in the legend the feature count of the features like in the following image:



- **RULE** set it to a given rule name to get only the named rule symbol
- **WIDTH/HEIGHT** the generated legend image size if the **RULE** parameter is set

3.1.5 GetProjectSettings

This request type works similar to **GetCapabilities**, but it is more specific to QGIS Server and allows a client to read additional information which is not available in the GetCapabilities output:

- initial visibility of layers
- information about vector attributes and their edit types
- information about layer order and drawing order
- list of layers published in WFS
- show if a group in the layer tree is mutually exclusive

3.2 Web Feature Service (WFS)

The **1.0.0** and **1.1.0** WFS standards implemented in QGIS Server provide a HTTP interface to query geographic features from a QGIS project. A typical WFS request defines the QGIS project to use and the layer to query.

Specifications document according to the version number of the service:

- [WFS 1.0.0](#)
- [WFS 1.1.0](#)

Standard requests provided by QGIS Server:

Request	Description
GetCapabilities	Returns XML metadata with information about the server
GetFeature	Returns a selection of features
DescribeFeatureType	Returns a description of feature types and properties
Transaction	Allows features to be inserted, updated or deleted

3.2.1 GetFeature

Standard parameters for the **GetFeature** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

Parameter	Required	Description
SERVICE	Yes	Name of the service
VERSION	No	Version of the service
REQUEST	Yes	Name of the request
TYPENAME	No	Name of layers
OUTPUTFORMAT	No	Output Format
RESULTTYPE	No	Type of the result
PROPERTYNAME	No	Name of properties to return
MAXFEATURES	No	Maximum number of features to return
SRSNAME	No	Coordinate reference system
FEATUREID	No	Filter the features by ids
FILTER	No	OGC Filter Encoding
BBOX	No	Map Extent
SORTBY	No	Sort the results

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
MAP	Yes	Specify the QGIS project file
STARTINDEX	No	Paging
GEOMETRYNAME	No	Type of geometry to return
EXP_FILTER	No	Expression filtering

SERVICE

This parameter has to be **WFS** in case of the **GetFeature** request.

For example:

```
http://localhost/qgisserver?
SERVICE=WFS
&...
```

VERSION

This parameter allows to specify the version of the service to use. Available values for the **VERSION** parameter are:

- 1.0.0
- 1.1.0

If no version is indicated in the request, then 1.1.0 is used by default.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&...
```


REQUEST

This parameter is `GetFeature` in case of the **GetFeature** request.

URL example:

```
http://localhost/qgisserver?  
SERVICE=WFS  
&VERSION=1.1.0  
&REQUEST=GetFeature  
&...
```

RESULTTYPE

This parameter may be used to specify the kind of result to return. Available values are:

- `results`: the default behavior
- `hits`: returns only a feature count

URL example:

```
http://localhost/qgisserver?  
SERVICE=WFS  
&VERSION=1.1.0  
&REQUEST=GetFeature  
&RESULTTYPE=hits  
&...
```

GEOMETRYNAME

This parameter can be used to specify the kind of geometry to return for features. Available values are:

- `extent`
- `centroid`
- `none`

URL example:

```
http://localhost/qgisserver?  
SERVICE=WFS  
&VERSION=1.1.0  
&REQUEST=GetFeature  
&GEOMETRYNAME=centroid  
&...
```

STARTINDEX

This parameter is standard in WFS 2.0, but it's an extension for WFS 1.0.0. Actually, it can be used to skip some features in the result set and in combination with `MAXFEATURES`, it provides the ability to page through results.

URL example:

```
http://localhost/qgisserver?  
SERVICE=WFS  
&VERSION=1.1.0  
&REQUEST=GetFeature  
&STARTINDEX=2  
&...
```

3.3 Web Map Tile Service (WMTS)

The **1.0.0** WMTS standard implemented in QGIS Server provides a HTTP interface to request tiled map images generated from a QGIS project. A typical WMTS request defines the QGIS project to use, some WMS parameters like layers to render, as well as tile parameters.

Specifications document of the service:

- [WMTS 1.0.0](#)

Standard requests provided by QGIS Server:

Request	Description
GetCapabilities	Returns XML metadata with information about the server
GetTile	Returns a tile
GetFeatureInfo	Retrieves data (geometry and values) for a pixel location

3.3.1 GetCapabilities

Standard parameters for the **GetCapabilities** request according to the OGC WMTS 1.0.0 specifications:

Parameter	Required	Description
SERVICE	Yes	Name of the service (WMTS)
REQUEST	Yes	Name of the request (GetCapabilities)

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
MAP	Yes	Specify the QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetCapabilities
&MAP=/home/qgis/projects/world.qgs
```

SERVICE

This parameter has to be `WMTS` in case of the **GetCapabilities** request.

REQUEST

This parameter is `GetCapabilities` in case of the **GetCapabilities** request.

MAP

This parameter allows to define the QGIS project file to use.

3.3.2 GetTile

Standard parameters for the **GetTile** request according to the OGC WMTS 1.0.0 specifications:

Parameter	Required	Description
SERVICE	Yes	Name of the service (WMTS)
REQUEST	Yes	Name of the request (GetTile)
LAYER	Yes	Layer identifier
FORMAT	Yes	Output format of the tile
TILEMATRIXSET	Yes	Name of the pyramid
TILEMATRIX	Yes	Meshing
TILEROW	Yes	Row coordinate in the mesh
TILECOL	Yes	Column coordinate in the mesh

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
MAP	Yes	Specify the QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetTile
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&FORMAT=image/png
&TILEMATRIXSET=EPSG:4326
&TILEROW=0
&TILECOL=0
```

SERVICE

This parameter has to be WMTS in case of the **GetTile** request.

REQUEST

This parameter is GetTile in case of the **GetTile** request.

LAYER

This parameter allows to specify the layer to display on the tile.

In addition, QGIS Server introduced some options to select a layer by:

- a short name
- the layer id

The short name of a layer may be configured through *Properties* ► *Metadata* in layer menu. If the short name is defined, then it's used by default instead of the layer's name:

```
http://localhost/qgisserver?  
SERVICE=WMTS  
&REQUEST=GetTile  
&LAYER=mynickname  
&...
```

Moreover, there's a project option allowing to select layers by their id in *OWS Server* ► *WMS capabilities* menu of the *Project* ► *Project Properties* dialog. To activate this option, the checkbox *Use layer ids as names* has to be selected.

```
http://localhost/qgisserver?  
SERVICE=WMTS  
&REQUEST=GetTile  
&LAYER=mylayerid1  
&...
```

FORMAT

This parameter may be used to specify the format of tile image. Available values are:

- jpg
- jpeg
- image/jpeg
- image/png

If the `FORMAT` parameter is different from one of these values, then the default format PNG is used instead.

TILEMATRIXSET

This parameter defines the CRS to use when computing the underlying pyramid. Format: `EPSG:XXXX`.

TILEMATRIX

This parameter allows to define the matrix to use for the output tile.

TILEROW

This parameter allows to select the row of the tile to get within the matrix.

TILECOL

This parameter allows to select the column of the tile to get within the matrix.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in *GetMap parameters table*, `MAP` is mandatory because a request needs a QGIS project to actually work. However, the `QGIS_PROJECT_FILE` environment variable may be used to define a default QGIS project. In this specific case, `MAP` is not longer a required parameter. For further information you may refer to *Advanced configuration*.

3.3.3 GetFeatureInfo

Standard parameters for the **GetFeatureInfo** request according to the OGC WMTS 1.0.0 specification:

Parameter	Required	Description
SERVICE	Yes	Name of the service (WMTS)
REQUEST	Yes	Name of the request (GetFeatureInfo)
LAYER	Yes	Layer identifier
INFOFORMAT	No	Output format
I	No	X coordinate of a pixel
J	No	Y coordinate of a pixel
TILEMATRIXSET	Yes	<i>See GetTile</i>
TILEMATRIX	Yes	<i>See GetTile</i>
TILEROW	Yes	<i>See GetTile</i>
TILECOL	Yes	<i>See GetTile</i>

In addition to the standard ones, QGIS Server supports the following extra parameters:

Parameter	Required	Description
MAP	Yes	Specify the QGIS project file

URL example:

```
http://localhost/qgisserver?
SERVICE=WMTS
&REQUEST=GetFeatureInfo
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&INFOFORMAT=image/html
&I=10
&J=5
```

SERVICE

This parameter has to be `WMTS` in case of the **GetFeatureInfo** request.

REQUEST

This parameter is `GetFeatureInfo` in case of the **GetFeatureInfo** request.

MAP

This parameter allows to define the QGIS project file to use.

As mentioned in *GetMap parameters table*, `MAP` is mandatory because a request needs a QGIS project to actually work. However, the `QGIS_PROJECT_FILE` environment variable may be used to define a default QGIS project. In this specific case, `MAP` is not longer a required parameter. For further information you may refer to *Advanced configuration*.

LAYER

This parameter allows to specify the layer to display on the tile.

In addition, QGIS Server introduced some options to select a layer by:

- a short name
- the layer id

The short name of a layer may be configured through *Properties* ► *Metadata* in layer menu. If the short name is defined, then it's used by default instead of the layer's name:

```
http://localhost/qgisserver?  
SERVICE=WMTS  
&REQUEST=GetFeatureInfo  
&LAYER=mynickname  
&...
```

Moreover, there's a project option allowing to select layers by their id in *OWS Server* ► *WMS capabilities* menu of the *Project* ► *Project Properties* dialog. To activate this option, the checkbox *Use layer ids as names* has to be selected.

```
http://localhost/qgisserver?  
SERVICE=WMTS  
&REQUEST=GetFeatureInfo  
&LAYER=mylayerid1  
&...
```

INFOFORMAT

This parameter allows to define the output format of the result. Available values are:

- text/xml
- text/html
- text/plain
- application/vnd.ogc.gml

The default value is text/plain.

I

This parameter allows to define the X coordinate of the pixel for which we want to retrieve underlying information.

J

This parameter allows to define the Y coordinate of the pixel for which we want to retrieve underlying information.

3.4 WFS3 (OGC API Features)

WFS3 is the first implementation of the new generation of OGC protocols. It is described by the [OGC API - Features - Part 1: Core](#) document.

Here is a quick informal summary of the most important differences between the well known WFS protocol and WFS3:

- WFS3 is based on a [REST API](#)
- WFS3 API must follow the [OPENAPI](#) specifications
- WFS3 supports multiple output formats but it does not dictate any (only GeoJSON and HTML are currently available in QGIS WFS3) and it uses [content negotiation](#) to determine which format is to be served to the client
- JSON and HTML are first class citizens in WFS3
- WFS3 is self-documenting (through the `/api` endpoint)
- WFS3 is fully navigable (through links) and browsable

Important: While the WFS3 implementation in QGIS can make use of the `MAP` parameter to specify the project file, no extra query parameters are allowed by the [OPENAPI](#) specification. For this reason it is strongly recommended that `MAP` is not exposed in the URL and the project file is specified in the environment by other means (i.e. setting `QGIS_PROJECT_FILE` in the environment through a web server rewrite rule).

Note: The [API](#) endpoint provides comprehensive documentation of all supported parameters and output formats of your service. The following paragraphs will only describe the most important ones.

3.4.1 Resource representation

The QGIS Server WFS3 implementation currently supports the following resource representation (output) formats:

- HTML
- JSON

The format that is actually served will depend on content negotiation, but a specific format can be explicitly requested by appending a format specifier to the endpoints.

Supported format specifier extensions are:

- `.json`
- `.html`

Additional format specifier aliases may be defined by specific endpoints:

- `.openapi`: alias for `.json` supported by the **API** endpoint
- `.geojson`: alias for `.json` supported by the **Features** and **Feature** endpoints

3.4.2 Endpoints

The API provides a list of endpoints that the clients can retrieve. The system is designed in such a way that every response provides a set of links to navigate through all the provided resources.

Endpoints points provided by the QGIS implementation are:

Name	Path	Description
Landing Page	/	General information about the service and provides links to all available endpoints
Conformance	/conformance	Information about the conformance of the service to the standards
API	/api	Full description of the endpoints provided by the service and the returned documents structure
Collections	/collections	List of all collections (i.e. 'vector layers') provided by the service
Collection	/collections/{collectionId}	Information about a collection (name, metadata, extent etc.)
Features	/collections/{collectionId}/items	List of the features provided by the collection
Feature	/collections/{collectionId}/items/{featureId}	Information about a single feature

Landing Page

The main endpoint is the **Landing Page**. From that page it is possible to navigate to all the available service endpoints. The **Landing Page** must provide links to

- the API definition (path `/api` link relations `service-desc` and `service-doc`),
- the Conformance declaration (path `/conformance`, link relation `conformance`), and
- the Collections (path `/collections`, link relation `data`).

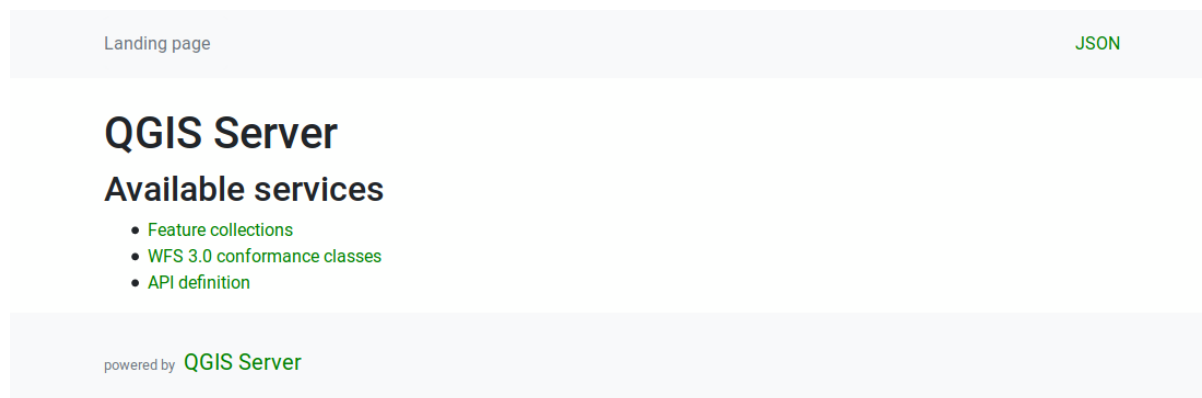


Fig. 3.2: Server WFS3 landing page

API Definition

The **API Definition** is an OPENAPI-compliant description of the API provided by the service. In its HTML representation it is a browsable page where all the endpoints and their response formats are accurately listed and documented. The path of this endpoint is `/api`.

The API definition provides a comprehensive and authoritative documentation of the service, including all supported parameters and returned formats.

Note: This endpoint is analogue to WFS's `GetCapabilities`

Collections list

The collections endpoint provides a list of all the collections available in the service. Since the service “serves” a single QGIS project the collections are the vector layers from the current project (if they were published as WFS in the project properties). The path of this endpoint is `/collections/`.

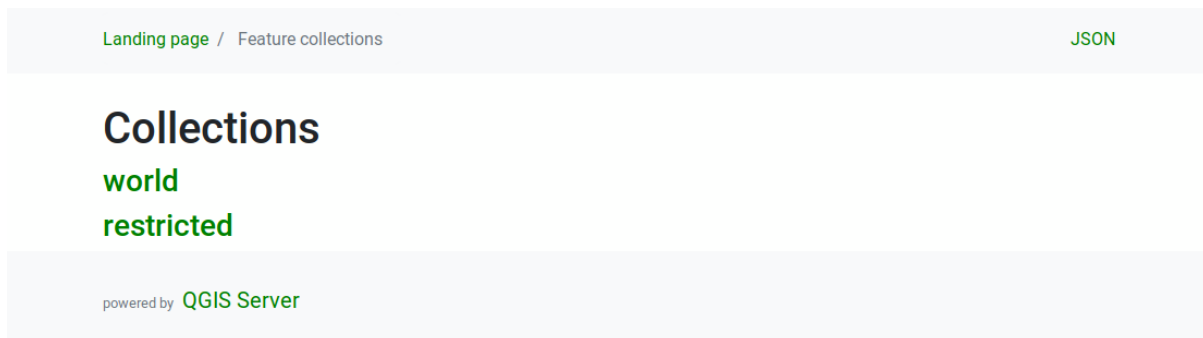


Fig. 3.3: Server WFS3 collections list page

Collection detail

While the collections endpoint does not provide detailed information about each available collection, that information is available in the `/collections/{collectionId}` endpoints. Typical information includes the extent, a description, CRSs and other metadata.

The HTML representation also provides a browsable map with the available features.

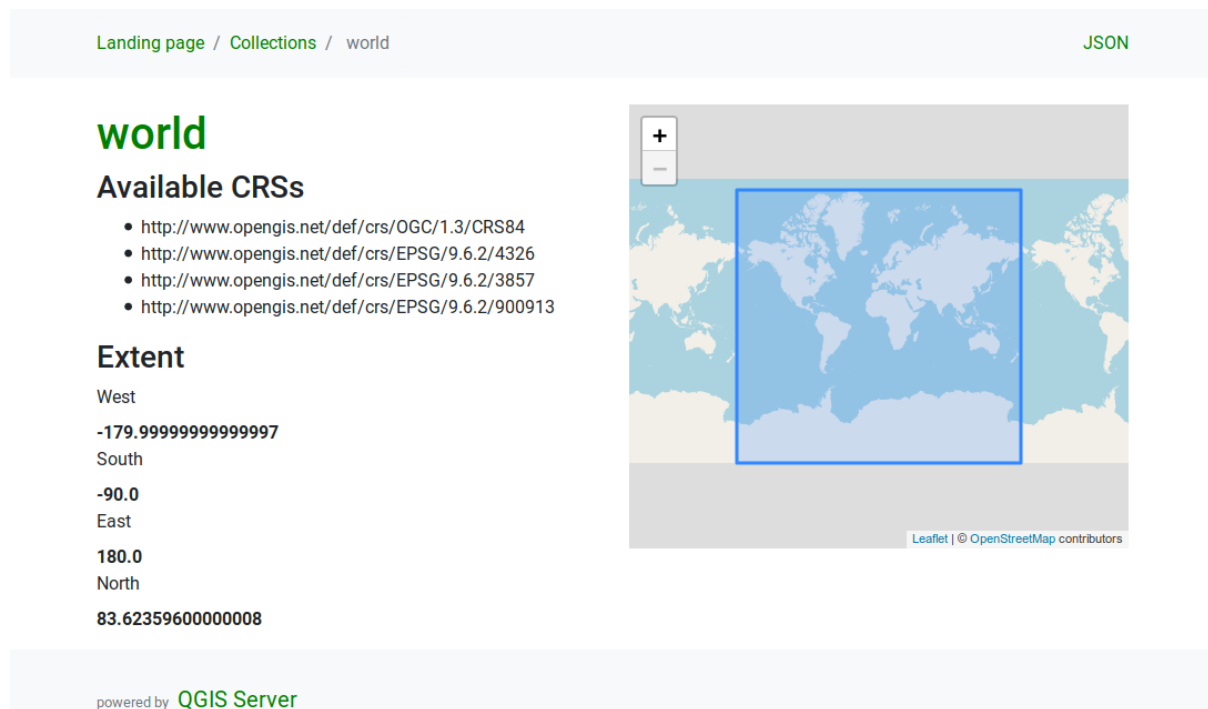


Fig. 3.4: Server WFS3 collection detail page

Features list

This endpoint provides a list of all features in a collection knowing the collection ID. The path of this endpoint is `/collections/{collectionId}/items`.

The HTML representation also provides a browsable map with the available features.

Note: This endpoint is analogue to `GetFeature` in WFS 1 and WFS 2.

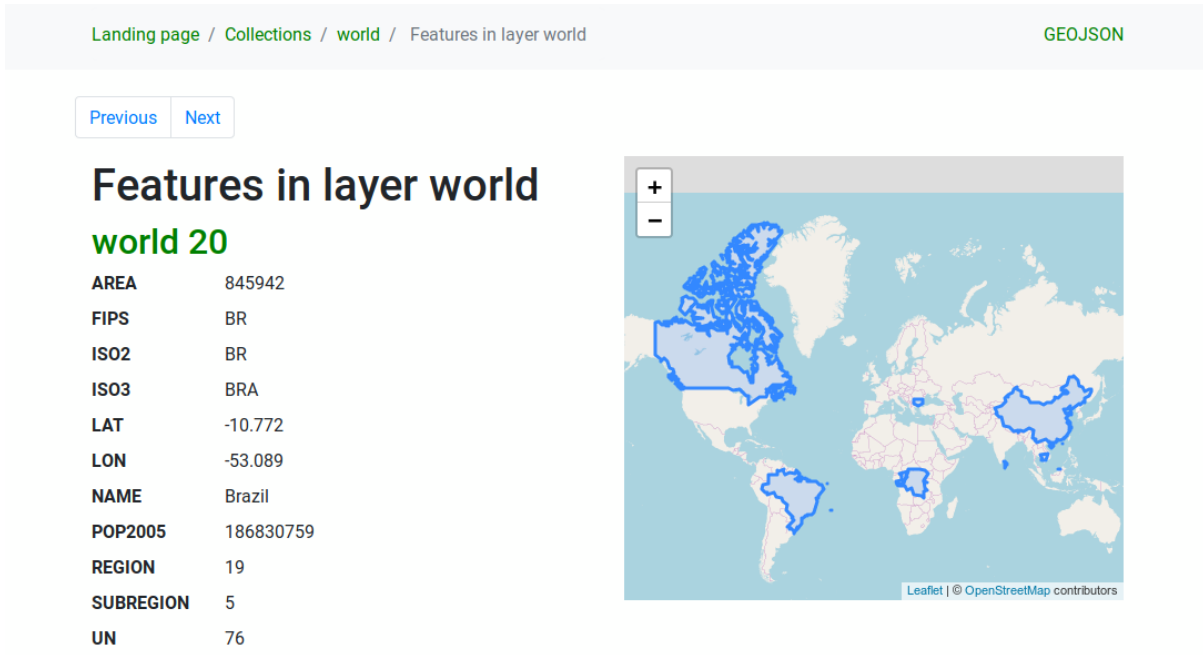


Fig. 3.5: Server WFS3 features list page

Feature detail

This endpoint provides all the available information about a single feature, including the feature attributes and its geometry. The path of this endpoint is `/collections/{collectionId}/items/{itemId}`.

The HTML representation also provides a browsable map with the feature geometry.



Fig. 3.6: Server WFS3 feature detail page

3.4.3 Pagination

Pagination of a long list of features is implemented in the OGC API through `next` and `prev` links, QGIS server constructs these links by appending `limit` and `offset` as query string parameters.

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?offset=10&limit=10
```

Note: The maximum acceptable value for `limit` can be configured with the `QGIS_SERVER_API_WFS3_MAX_LIMIT` server configuration setting (see: *Environment variables*).

3.4.4 Feature filtering

The features available in a collection can be filtered/searched by specifying one or more filters.

Date and time filter

Collections with date and/or datetime attributes can be filtered by specifying a `datetime` argument in the query string. By default the first date/datetime field is used for filtering. This behavior can be configured by setting a “Date” or “Time” dimension in the *QGIS Server* ► *Dimension* section of the layer properties dialog.

The date and time filtering syntax is fully described in the *API Definition* and also supports ranges (begin and end values are included) in addition to single values.

URL examples:

Returns only the features with date dimension matching 2019-01-01

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01
```

Returns only the features with datetime dimension matching 2019-01-01T01:01:01

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01T01:01:01
```

Returns only the features with datetime dimension in the range 2019-01-01T01:01:01 - 2019-01-01T12:00:00

```
http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01T01:01:01/2019-01-01T12:00:00
```

Bounding box filter

A bounding box spatial filter can be specified with the `bbox` parameter:

The order of the comma separated elements is:

- Lower left corner, WGS 84 longitude
- Lower left corner, WGS 84 latitude
- Upper right corner, WGS 84 longitude
- Upper right corner, WGS 84 latitude

Note: The OGC specifications also allow a 6 item bbox specifier where the third and sixth items are the Z components, this is not yet supported by QGIS server.

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=-180,-90,180,90
```

If the CRS of the bounding box is not **WGS 84**, a different CRS can be specified by using the optional parameter `bbox-crs`. The CRS format identifier must be in the **OGC URI** format:

URL example:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=913191,5606014,
↪913234,5606029&bbox-crs=http://www.opengis.net/def/crs/EPSG/9.6.2/3857
```

Attribute filters

Attribute filters can be combined with the bounding box filter and they are in the general form: `<attribute name>=<attribute value>`. Multiple filters can be combined using the AND operator.

URL example:

filters all features where attribute name equals “my value”

```
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=my%20value
```

Partial matches are also supported by using a * (“star”) operator:

URL example:

filters all features where attribute name ends with “value”

```
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=*value
```

3.4.5 Feature sorting

It is possible to order the result set by field value using the `orderby` query parameter.

The results are sorted in ascending order by default. To sort the results in descending order, a boolean flag (`sortdesc`) can be set:

```
http://localhost/qgisserver/wfs3/collection_one/items.json?orderby=name&sortdesc=1
```

3.4.6 Attribute selection

The feature attributes returned by a *Features list* call can be limited by adding a comma separated list of attribute names in the optional `properties` query string argument.

URL example:

returns only the name attribute

```
http://localhost/qgisserver/wfs3/collection_one/items.json?properties=name
```

3.4.7 Customize the HTML pages

The HTML representation uses a set of HTML templates to generate the response. The template is parsed by a template engine called `inja`. The templates can be customized by overriding them (see: *Template overrides*). The template has access to the same data that are available to the JSON representation and a few additional functions are available to the template:

Custom template functions

- `path_append(path)`: appends a directory path to the current url
- `path_chomp(n)`: removes the specified number “n” of directory components from the current url path
- `json_dump()`: prints the JSON data passed to the template
- `static(path)`: returns the full URL to the specified static path. For example: “`static(“/style/black.css”)`” with a root path “`http://localhost/qgisserver/wfs3`” will return “`http://localhost/qgisserver/wfs3/static/style/black.css`”.
- `links_filter(links, key, value)`: Returns filtered links from a link list
- `content_type_name(content_type)`: Returns a short name from a content type, for example “`text/html`” will return “HTML”

Template overrides

Templates and static assets are stored in subdirectories of the QGIS server default API resource directory (`/usr/share/qgis/resources/server/api/` on a Linux system), the base directory can be customized by changing the environment variable `QGIS_SERVER_API_RESOURCES_DIRECTORY`.

A typical Linux installation will have the following directory tree:

```

/usr/share/qgis/resources/server/api/
├── ogc
│   ├── schema.json
│   ├── static
│   │   ├── jsonFormatter.min.css
│   │   ├── jsonFormatter.min.js
│   │   └── style.css
│   └── templates
│       └── wfs3
│           ├── describeCollection.html
│           ├── describeCollections.html
│           ├── footer.html
│           ├── getApiDescription.html
│           ├── getFeature.html
│           ├── getFeatures.html
│           ├── getLandingPage.html
│           ├── getRequirementClasses.html
│           ├── header.html
│           ├── leaflet_map.html
│           └── links.html

```

To override the templates you can copy the whole tree to another location and point `QGIS_SERVER_API_RESOURCES_DIRECTORY` to the new location.

3.5 Extra parameters supported by all request types

The following extra parameters are supported by all protocols.

- **FILE_NAME**: if set, the server response will be sent to the client as a file attachment with the specified file name.

Note: Not available for WFS3.

- **MAP**: Similar to MapServer, the MAP parameter can be used to specify the path to the QGIS project file. You can specify an absolute path or a path relative to the location of the server executable (`qgis_mapserv.fcgi`). If not specified, QGIS Server searches for `.qgs` files in the directory where the server executable is located.

Example:

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?
  REQUEST=GetMap&MAP=/home/qgis/projects/world.qgs&...
```

Note: You can define a **QGIS_PROJECT_FILE** as an environment variable to tell the server executable where to find the QGIS project file. This variable will be the location where QGIS will look for the project file. If not defined it will use the MAP parameter in the request and finally look at the server executable directory.

3.6 REDLINING

This feature is available and can be used with `GetMap` and `GetPrint` requests.

The redlining feature can be used to pass geometries and labels in the request which are overlapped by the server over the standard returned image (map). This permits the user to put emphasis or maybe add some comments (labels) to some areas, locations etc. that are not in the standard map.

The `GetMap` request is in the format:

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi?map=/world.qgs&SERVICE=WMS&
  ↪VERSION=1.3.0&
  REQUEST=GetMap
  ...
  &HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, 2500000
  ↪5647000, 590000 5647000))
  &HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>
  ↪<FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter
  ↪name="stroke">%23ea1173</SvgParameter><SvgParameter name="stroke-opacity">1</
  ↪SvgParameter><SvgParameter name="stroke-width">1.6</SvgParameter></Stroke></
  ↪LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
  &HIGHLIGHT_LABELSTRING=Write label here
  &HIGHLIGHT_LABELSIZE=16
  &HIGHLIGHT_LABELCOLOR=%23000000
  &HIGHLIGHT_LABELBUFFERCOLOR=%23FFFFFF
  &HIGHLIGHT_LABELBUFFERSIZE=1.5
```

The `GetPrint` equivalent is in the format (note that `mapX:` parameter is added to tell which map has redlining):

```
http://qgisplatform.demo/cgi-bin/qgis_mapserv.fcgi?map=/world.qgs&SERVICE=WMS&
  ↪VERSION=1.3.0&
  REQUEST=GetPrint
  ...
```

(continues on next page)

(continued from previous page)

```
&map0:HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, ↵
↵2500000 5647000, 590000 5647000))
&map0:HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name>
↵<FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter ↵
↵name="stroke">%23ea1173</SvgParameter><SvgParameter name="stroke-opacity">1</
↵SvgParameter><SvgParameter name="stroke-width">1.6</SvgParameter></Stroke></
↵LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
&map0:HIGHLIGHT_LABELSTRING=Write label here
&map0:HIGHLIGHT_LABELSIZE=16
&map0:HIGHLIGHT_LABELCOLOR=%23000000
&map0:HIGHLIGHT_LABELBUFFERCOLOR=%23FFFFFF
&map0:HIGHLIGHT_LABELBUFFERSIZE=1.5
```

Here is the image outputted by the above request in which a polygon and a label are drawn on top of the normal map:



Fig. 3.7: Server response to a GetMap request with redlining parameters

You can see there are several parameters in this request:

- **HIGHLIGHT_GEOM:** You can add POINT, MULTILINESTRING, POLYGON etc. It supports multipart geometries. Here is an example: `HIGHLIGHT_GEOM=MULTILINESTRING((0 0, 0 1, 1 1))`. The coordinates should be in the CRS of the GetMap/GetPrint request.
- **HIGHLIGHT_SYMBOL:** This controls how the geometry is outlined and you can change the stroke width, color and opacity.
- **HIGHLIGHT_LABELSTRING:** You can pass your labeling text to this parameter.

- **HIGHLIGHT_LABELSIZE**: This parameter controls the size of the label.
- **HIGHLIGHT_LABELFONT**: This parameter controls the font of the label (e.g. Arial)
- **HIGHLIGHT_LABELCOLOR**: This parameter controls the label color.
- **HIGHLIGHT_LABELBUFFERCOLOR**: This parameter controls the label buffer color.
- **HIGHLIGHT_LABELBUFFERSIZE**: This parameter controls the label buffer size.

3.7 External WMS layers

QGIS Server allows including layers from external WMS servers in WMS GetMap and WMS GetPrint requests. This is especially useful if a web client uses an external background layer in the web map. For performance reasons, such layers should be directly requested by the web client (not cascaded via QGIS server). For printing however, these layers should be cascaded via QGIS server in order to appear in the printed map.

External layers can be added to the LAYERS parameter as EXTERNAL_WMS:<layername>. The parameters for the external WMS layers (e.g. url, format, dpiMode, crs, layers, styles) can later be given as service parameters <layername>:<parameter>. In a GetMap request, this might look like this:

```
http://localhost/qgisserver?
SERVICE=WMS&REQUEST=GetMap
...
&LAYERS=EXTERNAL_WMS:basemap,layer1,layer2
&STYLES=,,
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
&basemap:dpiMode=7
&basemap:crs=EPSG:2056
&basemap:layers=orthofoto
&basemap:styles=default
```

Similarly, external layers can be used in GetPrint requests:

```
http://localhost/qgisserver?
SERVICE=WMS
...
&REQUEST=GetPrint&TEMPLATE=A4
&map0:layers=EXTERNAL_WMS:basemap,layer1,layer2
&map0:EXTENT=<minx,miny,maxx,maxy>
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
&basemap:dpiMode=7
&basemap:crs=EPSG:2056
&basemap:layers=orthofoto
&basemap:styles=default
```

3.8 QGIS Server catalog

The QGIS Server Catalog is a simple catalog that shows the list of QGIS projects served by the QGIS Server. It provides a user-friendly fully browsable website with basic mapping capabilities to quickly browse the datasets exposed through those QGIS projects.

The QGIS Server catalog uses the variables QGIS_SERVER_LANDING_PAGE_PROJECTS_DIRECTORIES and QGIS_SERVER_LANDING_PAGE_PROJECTS_PG_CONNECTIONS (see *Environment variables*)

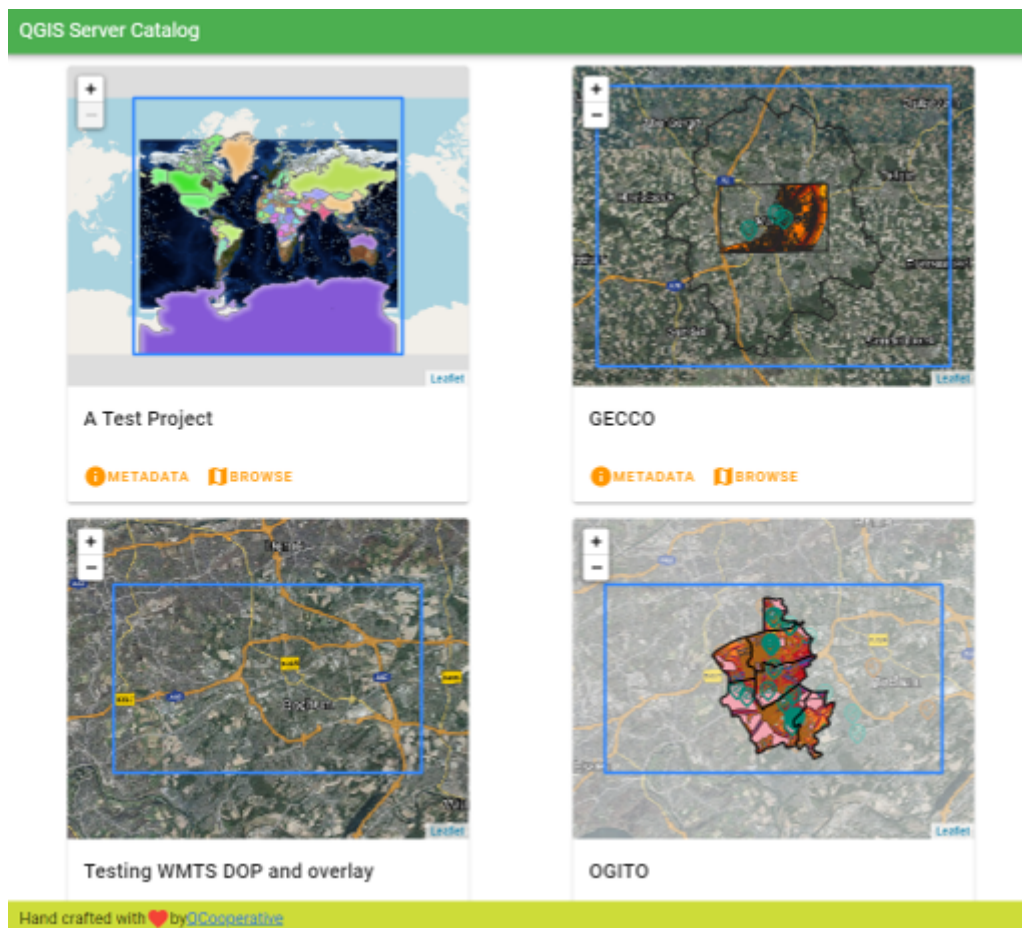


Fig. 3.8: Server Catalog project list page

You can consult the metadata associated to a project and the services that it provides. Links to those services are also given.

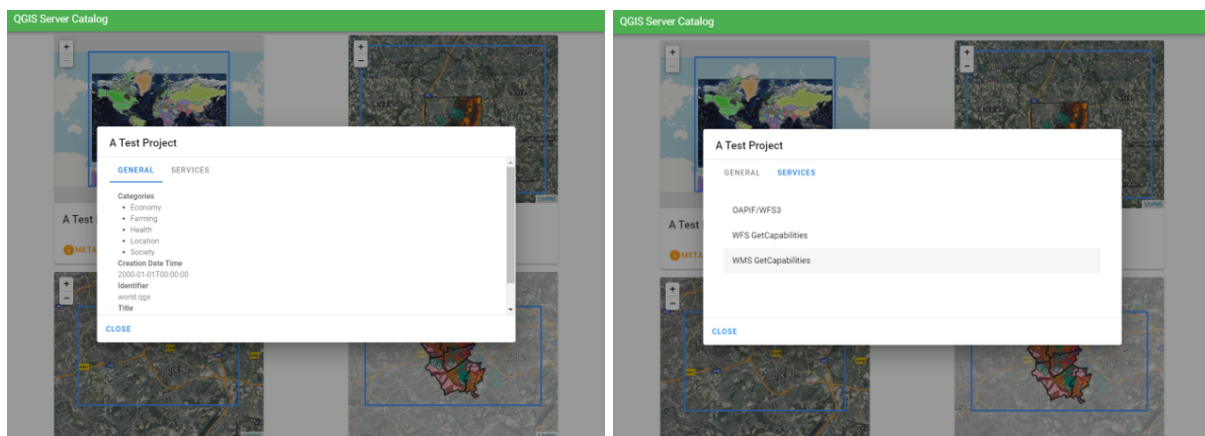


Fig. 3.9: Server Catalog, metadata associated to a project and services (links to) that it provides.

By browsing a project, it is listed the dataset that it serves.

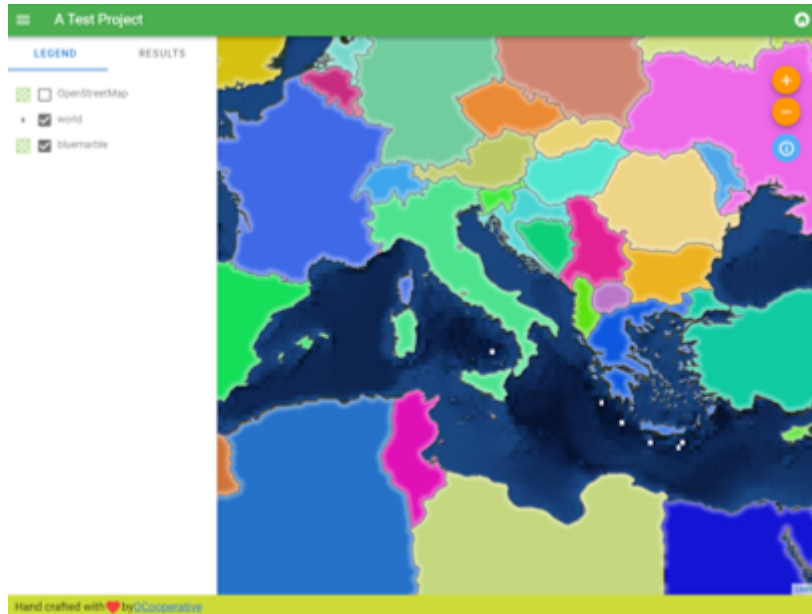


Fig. 3.10: Browsing a dataset served by a project in the Server Catalog

Use Right click on a layer to display the attribute table associated to it.

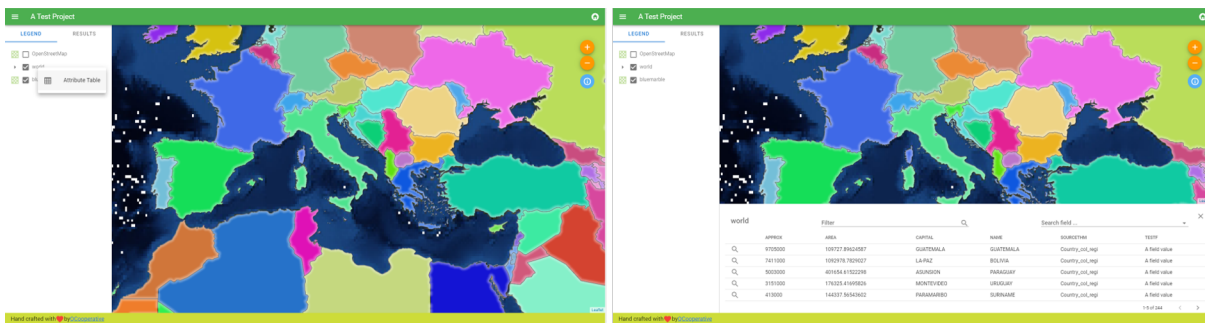


Fig. 3.11: Attribute table associated to a layer

It is possible to consult information of the elements in the map as shown in the image below:

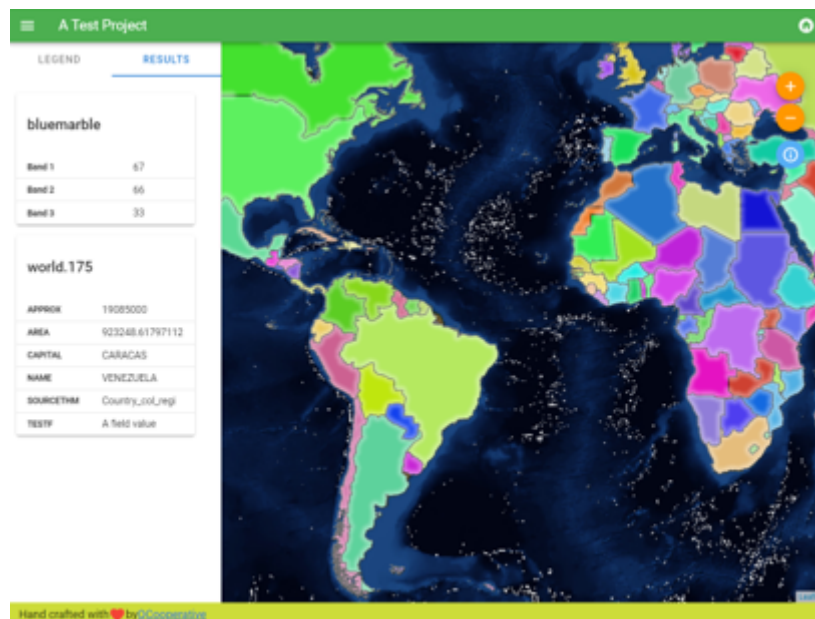


Fig. 3.12: Consulting information of a map element

4.1 Installation

To install the HelloWorld example plugin for testing the servers, you firstly have to create a directory to hold server plugins. This will be specified in the virtual host configuration and passed on to the server through an environment variable:

```
mkdir -p /var/www/qgis-server/plugins
cd /var/www/qgis-server/plugins
wget https://github.com/elpaso/qgis-helloserver/archive/master.zip
unzip master.zip
mv qgis-helloserver-master HelloServer
```

4.2 HTTP Server configuration

4.2.1 Apache

To be able to use server plugins, FastCGI needs to know where to look. So, we have to modify the Apache configuration file to indicate the **QGIS_PLUGINPATH** environment variable to FastCGI:

```
FcgidInitialEnv QGIS_PLUGINPATH "/var/www/qgis-server/plugins"
```

Moreover, a basic HTTP authorization is necessary to play with the HelloWorld plugin previously introduced. So we have to update the Apache configuration file a last time:

```
# Needed for QGIS HelloServer plugin HTTP BASIC auth
<IfModule mod_fcgid.c>
  RewriteEngine on
  RewriteCond %{HTTP:Authorization} .
  RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

Then, restart Apache:

```
systemctl restart apache2
```

4.3 How to use a plugin

Test the server with the HelloWorld plugin:

```
wget -q -O - "http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=HELLO"  
HelloServer!
```

You can have a look at the default GetCapabilities of the QGIS server at:

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&  
↪REQUEST=GetCapabilities
```

ADVANCED CONFIGURATION

5.1 Logging

To log requests sent to the server, you have to set the following environment variable:

- *QGIS_SERVER_LOG_STDERR*

With the following variables the logging can be further customized:

- *QGIS_SERVER_LOG_LEVEL*
- *QGIS_SERVER_LOG_PROFILE*

5.2 Environment variables

You can configure some aspects of QGIS Server by setting **environment variables**.

According to the HTTP server and how you run QGIS Server, there are several ways to define these variables. This is fully described in *Apache HTTP Server*.

Name	Description	Default	Services
QGIS_OPTIONS_PATH	Specifies the path to the directory with settings. It works the same way as QGIS application <code>--optionspath</code> option. It is looking for settings file in <code><QGIS_OPTIONS_PATH>/QGIS/QGIS3.ini</code> .	"	All
QGIS_PLUGINPATH	Useful if you are using Python plugins for the server, this sets the folder that is searched for Python plugins.	"	All
QGIS_PROJECT_FILE	The <code>.qgs</code> or <code>.qgz</code> project file, normally passed as a parameter in the query string (with <i>MAP</i>), you can also set it as an environment variable (for example by using <code>mod_rewrite</code> Apache module). Note that you may also indicate a project stored in PostgreSQL, e.g. <code>postgresql://localhost:5432?sslmode=disable&dbname=mydb&schema=myschema&project=myproject</code> .	"	All
QGIS_SERVER_API_RESOURCE_DIRECTORY	Base directory for all OGC API (such as OAPIF/WFS3) static resources (HTML templates, CSS, JS, ...)	depends on packaging	WFS
QGIS_SERVER_API_WFS3_MAXIMUM_FEATURES	Maximum value for <code>limit</code> in a features request.	10000	WFS
QGIS_SERVER_CACHE_DIRECTORY	Specifies the network cache directory on the filesystem.	cache in profile directory	All
QGIS_SERVER_CACHE_SIZE	Sets the network cache size in MB.	50 MB	All
QGIS_SERVER_DISABLE_GETPRINT	Provides an option at the project level to improve project read time by disabling loading of layouts. Activating this option disables the QGIS WMS GetPrint request. Set this QGIS project flag to not load layouts.	false	WMS
QGIS_SERVER_IGNORE_BAD_LAYERS	Bad layers are layers that cannot be loaded. The default behavior of QGIS Server is to consider the project as not available if it contains a bad layer. The default behavior can be overridden by setting this variable to <code>1</code> or <code>true</code> . In this case, "bad" layers will just be ignored, and the project will be considered valid and available.	false	All
QGIS_SERVER_LANDING_PAGE_PROJECTS_DIRECTORIES	Directories used by the landing page service to find <code>.qgs</code> and <code>.qgz</code> projects	<code>/qgis/server/projects_directories</code>	All
QGIS_SERVER_LANDING_PAGE_PROJECTS_PG_CONNECTIONS	PostgreSQL connection strings used by the landing page service to find projects	<code>/qgis/server/projects_pg_connections</code>	All
QGIS_SERVER_LOG_FILE	Specify path and filename. Make sure that server has proper permissions for writing to file. File should be created automatically, just send some requests to server. If it's not there, check permissions.	"	All

Warning:
QGIS_SERVER_LOG_FILE
 is deprecated since QGIS 3.4, use

5.3 Settings summary

When QGIS Server is starting, you have a summary of all configurable parameters thanks to environment variables. Moreover, the value currently used and the origin is also displayed.

For example with spawn-fcgi:

```
export QGIS_OPTIONS_PATH=/home/user/.local/share/QGIS/QGIS3/profiles/default/
export QGIS_SERVER_LOG_STDERR=1
export QGIS_SERVER_LOG_LEVEL=2
spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi -s /tmp/qgisserver.sock -U www-
↳data -G www-data -n

QGIS Server Settings:

- QGIS_OPTIONS_PATH / '' (Override the default path for user configuration): '/
↳home/user/.local/share/QGIS/QGIS3/profiles/default/' (read from ENVIRONMENT_
↳VARIABLE)

- QGIS_SERVER_PARALLEL_RENDERING / '/qgis/parallel_rendering' (Activate/
↳Deactivate parallel rendering for WMS getMap request): 'true' (read from INI_
↳FILE)

- QGIS_SERVER_MAX_THREADS / '/qgis/max_threads' (Number of threads to use when
↳parallel rendering is activated): '4' (read from INI_FILE)

- QGIS_SERVER_LOG_LEVEL / '' (Log level): '2' (read from ENVIRONMENT_VARIABLE)

- QGIS_SERVER_LOG_STDERR / '' (Activate/Deactivate logging to stderr): '1'
↳(read from ENVIRONMENT_VARIABLE)

- QGIS_PROJECT_FILE / '' (QGIS project file): '' (read from DEFAULT_VALUE)

- MAX_CACHE_LAYERS / '' (Specify the maximum number of cached layers): '100'
↳(read from DEFAULT_VALUE)

- QGIS_SERVER_CACHE_DIRECTORY / '/cache/directory' (Specify the cache
↳directory): '/root/.local/share/QGIS/QGIS3/profiles/default/cache' (read from
↳DEFAULT_VALUE)

- QGIS_SERVER_CACHE_SIZE / '/cache/size' (Specify the cache size): '52428800'
↳(read from INI_FILE)

Ini file used to initialize settings: /home/user/.local/share/QGIS/QGIS3/profiles/
↳default/QGIS/QGIS3.ini
```

In this particular case, we know that **QGIS_SERVER_MAX_THREADS** and **QGIS_SERVER_PARALLEL_RENDERING** values are read from the ini file found in **QGIS_OPTIONS_PATH** directory (which is defined through an environment variable). The corresponding entries in the ini file are **/qgis/max_threads** and **/qgis/parallel_rendering** and their values are **true** and **4** threads.

5.4 Short name for layers, groups and project

A number of elements have both a <Name> and a <Title>. The **Name** is a text string used for machine-to-machine communication while the **Title** is for the benefit of humans.

For example, a dataset might have the descriptive Title “Maximum Atmospheric Temperature” and be requested using the abbreviated **Name** “ATMAX”. The user can set the title for layers, groups and projects.

OWS name is based on the name used in the layer tree. This name is more a label for humans than a name for machine-to-machine communication. You can set a **Short name** for layers, groups or projects, to be used by QGIS Server as the layer identification name (in *LAYERS* parameter for instance).

You can set title, short name and abstract for:

- **Layers:** right-click on a layer and choose *Properties... ► QGIS Server ► Description*.
- **Groups:** right-click on a group and select *Set Group WMS data*

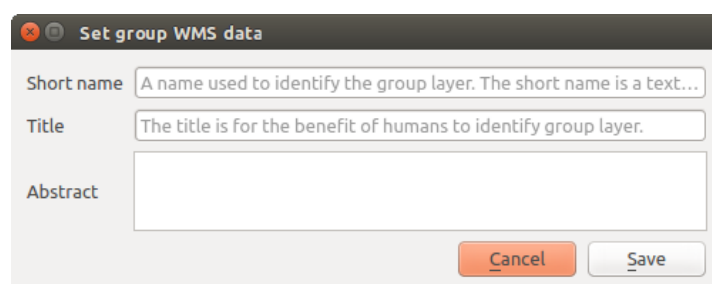


Fig. 5.1: Set group WMS data dialog

- **Project:** go to *Project ► Properties... ► QGIS Server ► Service Capabilities*.

5.5 Connection to service file

In order to make apache aware of the PostgreSQL service file (see the pg-service-file section) you need to make your *.conf file look like:

```
SetEnv PGSERVICEFILE /home/web/.pg_service.conf
<Directory "/home/web/apps2/bin/">
  AllowOverride None
  . . . . .
```

5.6 Add fonts to your linux server

Keep in mind that you may use QGIS projects that point to fonts that may not exist by default on other machines. This means that if you share the project, it may look different on other machines (if the fonts don't exist on the target machine).

In order to ensure this does not happen you just need to install the missing fonts on the target machine. Doing this on desktop systems is usually trivial (double clicking the fonts).

For linux, if you don't have a desktop environment installed (or you prefer the command line) you need to:

- On Debian based systems:

```
sudo su
mkdir -p /usr/local/share/fonts/truetype/myfonts && cd /usr/local/share/fonts/
↳truetype/myfonts

# copy the fonts from their location
cp /fonts_location/* .

chown root *
cd .. && fc-cache -f -v
```

- On Fedora based systems:

```
sudo su
mkdir /usr/share/fonts/myfonts && cd /usr/share/fonts/myfonts

# copy the fonts from their location
cp /fonts_location/* .

chown root *
cd .. && fc-cache -f -v
```


DEVELOPMENT SERVER

A production installation and deployment of QGIS Server usually involves setting up a web server component (e.g. Apache or Nginx) that can forward the HTTP requests coming from the clients to the QGIS Server FastCGI binary application.

If you want to quickly test QGIS Server on your local machine without configuring and installing a full web server stack you can use the QGIS Development Standalone server.

This is an independent application that provides a very simple web server ready to serve your project files.

Warning: The Standalone Development Server has not been developed with the purpose of being used in production, it was not checked for security vulnerabilities or for other stress conditions that normally will occur on a publicly exposed server.

To launch the server:

```
$ qgis_mapserver
```

The default port the Development Server listens to is 8000. Example output:

```
QGIS Development Server listening on http://localhost:8000
CTRL+C to exit
127.0.0.1 [lun gen 20 15:16:41 2020] 5140 103ms "GET /wfs3/?MAP=/tests/testdata/
↳qgis_server/test_project.qgs HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 3298 2ms "GET /wfs3/static/jsonFormatter.min.
↳js HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1678 3ms "GET /wfs3/static/jsonFormatter.min.
↳css HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1310 5ms "GET /wfs3/static/style.css HTTP/1.1
↳" 200
127.0.0.1 [lun gen 20 15:16:43 2020] 4285 13ms "GET /wfs3/collections?MAP=/tests/
↳testdata/qgis_server/test_project.qgs HTTP/1.1" 200
```

The server has a few options that can be passed as command line arguments. You can see them all by invoking the server with `-h`.

```
Usage: qgis_mapserver [options] [address:port]
QGIS Development Server

Options:
-h, --help           Displays this help.
-v, --version       Displays version information.
-l <logLevel>      Sets log level (default: 0)
                    0: INFO
                    1: WARNING
                    2: CRITICAL
-p <projectPath>   Path to a QGIS project file (*.qgs or *.qgz),
                    if specified it will override the query string MAP argument
```

(continues on next page)

(continued from previous page)

and the `QGIS_PROJECT_FILE` environment variable

Arguments:

`addressAndPort` Listen to address and port (default: `"localhost:8000"`)
address and port can also be specified with the environment
variables `QGIS_SERVER_ADDRESS` and `QGIS_SERVER_PORT`

CONTAINERIZED DEPLOYMENT

There are many ways to use containerized application, from the most simple (simple Docker images) to sophisticated (Kubernetes and so on).

Note: This kind of deployment needs the [docker application](#) to be installed and running. Check this [tutorial](#).

Hint: Docker run pre packaged application (aka images) which can be retrieved as sources (Dockerfile and resources) to build or already built from registries (private or public).

Note: QGIS Debian-Ubuntu package downloads need a valid gpg authentication key. Please refer to the [installation pages](#) to update the following Dockerfile with the latest key fingerprint

7.1 Simple docker images

As the docker image does not exist in a public registry. you will need to build it. To do so create a directory `qgis-server` and within its directory:

- create a file `Dockerfile` with this content:

```
FROM debian:buster-slim

ENV LANG=en_EN.UTF-8

RUN apt-get update \
    && apt-get install --no-install-recommends --no-install-suggests --allow-
↳unauthenticated -y \
    gnupg \
    ca-certificates \
    wget \
    locales \
    && localedef -i en_US -f UTF-8 en_US.UTF-8 \
    # Add the current key for package downloading - As the key changes every year.
↳at least
    # Please refer to QGIS install documentation and replace it with the latest one
    && wget -O - https://qgis.org/downloads/qgis-2020.gpg.key | gpg --import \
    && gpg --export --armor F7E06F06199EF2F2 | apt-key add - \
    && echo "deb http://qgis.org/debian buster main" >> /etc/apt/sources.list.d/
↳qgis.list \
    && apt-get update \
    && apt-get install --no-install-recommends --no-install-suggests --allow-
↳unauthenticated -y \
```

(continues on next page)

```

    qgis-server \
    spawn-fcgi \
    xauth \
    xvfb \
    && apt-get remove --purge -y \
    gnupg \
    wget \
    && rm -rf /var/lib/apt/lists/*

RUN useradd -m qgis

ENV TINI_VERSION v0.17.0
ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/tini /tini
RUN chmod +x /tini

ENV QGIS_PREFIX_PATH /usr
ENV QGIS_SERVER_LOG_STDERR 1
ENV QGIS_SERVER_LOG_LEVEL 2

COPY cmd.sh /home/qgis/cmd.sh
RUN chmod -R 777 /home/qgis/cmd.sh
RUN chown qgis:qgis /home/qgis/cmd.sh

USER qgis
WORKDIR /home/qgis

ENTRYPOINT ["/tini", "--"]

CMD ["/home/qgis/cmd.sh"]

```

- create a file `cmd.sh` with this content:

```

#!/bin/bash

[[ $DEBUG == "1" ]] && env

exec /usr/bin/xvfb-run --auto-servernum --server-num=1 /usr/bin/spawn-fcgi -p 5555 \
  ↪-n -d /home/qgis -- /usr/lib/cgi-bin/qgis_mapserv.fcgi

```

- build the image with:

```

docker build -f Dockerfile -t qgis-server ./

```

7.1.1 First run

To run the server you will need a QGIS project file. You can use one of yours or pick [this sample](#).

To do so, create a directory `data` within the directory `qgis-server` and copy your file in it. To comply with the following explanations, rename it to `osm.qgs`.

Note: You may need to add advertised URLs under the *QGIS Server* tab of the *Project ► Properties* if the Get-Capabilities are broken. For example if your server is exposed on port 8080, you will put this for advertised URL `http://localhost:8080/qgis-server/`. More information available in section *Configure your project* and subsequent.

Now, you can run the server with:


```
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
```

Options used:

- **-d**: run in the background
- **--rm**: remove the container when it is stopped
- **--name**: name of the container to be created
- **--net**: (previously created) sub network
- **--hostname**: container hostname, for later referencing
- **-v**: local data directory to be mounted in the container
- **-p**: host/container port mapping
- **-e**: environment variable to be used in the container

To check, type `docker ps | grep qgis-server` and you should see a line with **qgis-server**:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	
↔ PORTS		NAMES			
4de8192da76e	qgis-server	"/tini -- /home/qgis..."	3 seconds ago	Up 2 seconds	↔
↔ 0.0.0.0:5555->5555/tcp		qgis-server			

7.1.2 Usable sample

As the server is only accepting fastcgi connections, you need an HTTP server that handles this protocol. To do so we have to create a simple Nginx configuration file and start a Nginx image.

Create a file `nginx.conf` in the current directory with this content:

```
server {
    listen 80;
    server_name _;
    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
    }
    location /qgis-server {
        proxy_buffers 16 16k;
        proxy_buffer_size 16k;
        gzip off;
        include fastcgi_params;
        fastcgi_pass qgis-server:5555;
    }
}
```

And type this command:

```
docker run -d --rm --name nginx --net=qgis --hostname=nginx \
-v $(pwd)/nginx.conf:/etc/nginx/conf.d/default.conf:ro -p 8080:80 \
nginx:1.13
```

To check capabilities availability, type in a browser <http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities>

7.1.3 Cleanup

To cleanup the running images, type:

```
docker stop qgis-server nginx
```

7.2 Docker stacks

The previous method is scriptable, but not easily packageable nor standardized or easily manageable.

To work with a docker image set you could use a docker stack managed by an orchestrator. In a stack, the images are working in the same private network, and you can start / stop the whole stack or deploy the stack to other workers. There are many orchestrators, for example Swarm, Kubernetes and Mesos.

In the following, we will present simple configurations for testing purposes. They are not suitable for production.

7.2.1 Swarm/docker-compose

Docker now has its own orchestrator: Swarm (compatible with docker-compose files). You have to [enable it](#) (the Mac version will also work with Linux).

Stack description

Now that you have Swarm working, create the service file (see [Deploy to Swarm](#)) `qgis-stack.yml`:

```
version: '3.7'

services:
  qgis-server:
    # Should use version with utf-8 locale support:
    image: qgis-server:latest
    volumes:
      - REPLACE_WITH_FULL_PATH/data:/data:ro
    environment:
      - LANG=en_EN.UTF-8
      - QGIS_PROJECT_FILE=/data/osm.qgs
      - QGIS_SERVER_LOG_LEVEL=0 # INFO (log all requests)
      - DEBUG=1 # display env before spawning QGIS Server

  nginx:
    image: nginx:1.13
    ports:
      - 8080:80
    volumes:
      - REPLACE_WITH_FULL_PATH/nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - qgis-server
```

To deploy (or update) the stack, type:

```
docker stack deploy -c qgis-stack.yml qgis-stack
```

Check the stack deployment status until you obtain **1/1** in the **replicas** column:

```
docker stack services qgis-stack
```

Something like:

ID	NAME	MODE	REPLICAS	
↔IMAGE	PORTS			
gmx7ewlvwsqt	qgis_nginx	replicated	1/1	
↔nginx:1.13	*:8080->80/tcp			
10v2e7cl43u3	qgis_qgis-server	replicated	1/1	
↔qgis-server:latest				

To check WMS capabilities, type in a web browser <http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities>

Cleanup

To cleanup, type:

```
docker stack rm qgis-stack
```

7.2.2 Kubernetes

Installation

If you have a **Docker Desktop** installation, using Kubernetes (aka k8s) is pretty straight forward: [enable k8s](#).

If not, follow the [minikube tutorial](#) or [microk8s for Ubuntu](#).

As Kubernetes installation can be really complex, we will only focus on aspects used by this demo. For further / deeper information, check the [official documentation](#).

microk8s

microk8s needs extra steps: you have to enable the registry and tag the qgis-server image in order to have Kubernetes to find the created images.

First, enable the registry:

```
microk8s enable dashboard dns registry
```

Then, tag and push the image to your newly created registry:

```
docker tag qgis-server 127.0.0.1:32000/qgis-server && docker push 127.0.0.1:32000/↔qgis-server
```

Finally, add or complete the `/etc/docker/daemon.json` to have your registry **127.0.0.1:32000** listed in the **insecure-registries** field:

```
{
  "insecure-registries": ["127.0.0.1:32000"]
}
```

Creating manifests

Kubernetes describes the objects to deploy in yaml manifests. There are many different kinds, but we will only use deployments (handle pods, i.e. docker images) and services to expose the deployments to internal or external purposes.

Deployment manifests

Create a file `deployments.yaml` with this content:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-server
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-server
  template:
    metadata:
      labels:
        myLabel: qgis-server
    spec:
      containers:
        - name: qgis-server
          image: localhost:32000/qgis-server:latest
          imagePullPolicy: IfNotPresent
          env:
            - name: LANG
              value: en_EN.UTF-8
            - name: QGIS_PROJECT_FILE
              value: /data/osm.qgs
            - name: QGIS_SERVER_LOG_LEVEL
              value: "0"
            - name: DEBUG
              value: "1"
          ports:
            - containerPort: 5555
          volumeMounts:
            - name: qgis-data
              mountPath: /data/
          volumes:
            - name: qgis-data
              hostPath:
                path: REPLACE_WITH_FULL_PATH/data
    ---
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: qgis-nginx
    namespace: default
  spec:
    replicas: 1
    selector:
      matchLabels:
        myLabel: qgis-nginx
    template:
      metadata:

```

(continues on next page)

(continued from previous page)

```

labels:
  myLabel: qgis-nginx
spec:
  containers:
    - name: qgis-nginx
      image: nginx:1.13
      ports:
        - containerPort: 80
      volumeMounts:
        - name: nginx-conf
          mountPath: /etc/nginx/conf.d/default.conf
  volumes:
    - name: nginx-conf
      hostPath:
        path: REPLACE_WITH_FULL_PATH/nginx.conf

```

Service manifests

Create a file `services.yaml` with this content:

```

apiVersion: v1
kind: Service
metadata:
  name: qgis-server
  namespace: default
spec:
  type: ClusterIP
  selector:
    myLabel: qgis-server
  ports:
    - port: 5555
      targetPort: 5555
---
apiVersion: v1
kind: Service
metadata:
  name: qgis-nginx
  namespace: default
spec:
  type: NodePort
  selector:
    myLabel: qgis-nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080

```

Deploying manifests

To deploy the images and services in Kubernetes, one can use the dashboard (click on the + on the upper right) or the command line.

Note: When using the command line with `microk8s` you will have to prefix each command with `microk8s`.

To deploy or update your manifests:

```
kubectl apply -k ./
```

To check what is currently deployed:

```
kubectl get pods, services, deployment
```

You should obtain something like:

NAME	READY	STATUS	RESTARTS	AGE
pod/qgis-nginx-54845ff6f6-8skp9	1/1	Running	0	27m
pod/qgis-server-75df8ddd89-c7t7s	1/1	Running	0	27m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	
↪ AGE					
service/Kubernetes	ClusterIP	10.152.183.1	<none>	443/TCP	↪
↪ 5h51m					
service/qgis-exec-server	ClusterIP	10.152.183.218	<none>	5555/TCP	↪
↪ 35m					
service/qgis-nginx	NodePort	10.152.183.234	<none>	80:30080/TCP	↪
↪ 27m					
service/qgis-server	ClusterIP	10.152.183.132	<none>	5555/TCP	↪
↪ 27m					

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/qgis-nginx	1/1	1	1	27m
deployment.apps/qgis-server	1/1	1	1	27m

To read nginx/qgis logs, type:

```
kubectl logs -f POD_NAME
```

To check WMS capabilities, type in a web browser <http://localhost:30080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities>

Cleanup

To clean up, type:

```
kubectl delete -n default service/qgis-server service/qgis-nginx deployment/qgis-  
↪nginx deployment/qgis-server
```

7.3 Cloud deployment

Managing your own cluster of servers to handle the deployment of containerized applications, is a complex job. You have to handle multiple issues, such as hardware, bandwidth and security at different levels.

Cloud deployment solutions can be a good alternative when you do not want to focus on infrastructure management.

A cloud deployment may use proprietary mechanisms, but they are also compatible with the stages explained previously (*docker images* and *stack management*).

7.3.1 AWS usecase

With Amazon AWS, through [ECS \(Elastic Container Service\)](#) functionalities, you can use `docker-compose` or Kubernetes compatible wrappers to manage your stack. You will have to create an [image registry](#) for your custom images to be accessible.

To use `docker-compose` alike functionalities, you need to install the `ecs-cli` client and have [proper permissions / roles](#). Then, with the help of the `ecs-cli compose` commands (see the [ecs-cli compose manual](#) and [ecs-cli tutorial](#)), you can reuse the [stack description](#).

To use Kubernetes, you can use the AWS web console or the command line tool `eksctl` and have the [proper permissions / roles](#). Then with a well configured `kubectl` environment, you can reuse the [Kubernetes manifests](#).

FREQUENTLY ASKED QUESTION

- *What are the differences between QGIS Desktop and QGIS Server?*

QGIS Desktop has a graphical user interface and allows you to create and modify maps. QGIS Server is a server application serving your QGIS project files to end user applications via OGC web services like WMS, WFS, etc..

- *What is OGC?*

The OGC (Open Geospatial Consortium) is an international not for profit organization committed to making quality open standards for the global geospatial community.

- *Name some other web mapping servers?*

ArcGIS server, Geoserver, Mapserver, Mapnik etc.

- *How to compare QGIS server to other web mapping servers? (2021/01/01)*

Features	QGIS Server	GeoServer	ArcGIS Server
Since	2006	2001	1999
Licence	GPL	GPL	commercial
Commercial support	Multiple companies	Multiple companies	ESRI and its vendors network
Technology	C++/python	Java	C++
Tile cache	yes	yes (via GeoWebCache)	yes
3D	No	No	Yes
Querying	FES (2.0) and OGC (1.0) filters	CQL and OGC filters	OGC filters
Report generation	yes	yes	yes
Server administration	yes via third parties (LizMap, QWC2, etc.)	web + API REST	web + API REST
GIS project Layer/symbology edition	complete via dedicated GUI	simple via web interface	complete via dedicated GUI

- *What are the OGC specification versions implemented in QGIS server compared to other web mapping servers? (2021/01/01)*

OGC standards	QGIS Server	GeoServer	ArcGIS Server
WMS (Web Map Service)	1.3.0 - 1.1.1	1.3.0 - 1.1.1	1.3.0 - 1.1.1
WFS (Web Feature Service)	1.1.0 - 1.0.0	2.0.0 - 1.0.0	2.0.0 - 1.0.0
OAPIF (aka WFS3)	1.0.0	no	no
WMTS (Web Map Tile Service)	1.0.0	1.0.0	1.0.0
WCS (Web Coverage Service)	1.0.0	2.0.1 - 1.0.0	2.0.1 - 1.0.0
WPS (Web Processing Service)	no	1.0.0	1.0.0
CSW (Catalogue Service for the Web)	no	2.0.2	no
SLD (Styled Layer Descriptor)	yes	yes	yes

- *What is a tile cache?*

Maps are often static. As most mapping clients render WMS (Web Map Service) data every time they are queried, this can result in unnecessary processing and increased wait times.

The tile cache optimizes this experience by saving (caching) map images, or tiles, as they are requested, in effect acting as a proxy between client (such as OpenLayers or Google Maps) and server (any WMS-compliant server). As new maps and tiles are requested, QGIS server intercepts these calls and returns pre-rendered tiles if stored, or calls the QGIS engine to render new tiles as necessary. Thus, once tiles are stored, the speed of map rendering increases by many times, creating a much improved user experience.

- *What is PostgreSQL?*

PostgreSQL is a powerful, open source object-relational database companion for QGIS.

- *What is PostGIS?*

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.

- To be continued...