pg_featureserv

pg_tileserv

A simpler GIS architecture

whoami

- Augustin Trancart (@autra42 on twitter)
- Sr GIS Engineer @ Oslandia

Before

typical GIS web solution:

- map served as raster to client
- rendering engine needed (server side)

then came Vector Tiles

(and client side rendering)

- client-side rendering has many advantages
    - dynamic styling (on hover/click/...)
    - picking / metadata
    - offloading rendering load (where it makes most sense)
    - accessibility?
- Do we still need {qgis,geo,map}server?
- -> datasources connexion, http servers, OGC protocol compatibility...

Idea under these projects

- postgis can output vector data and geojson
- front client knows how to read it
- Missing part: connexion to PostGIS + standard protocol + some bonuses

Common points

- written in go
- REST api
- Postgis only
- (just configure the db connection)
- can serve tables, views, and functions!

pg_tileserv

- protocol: xyz tile system
- format: Mapbox Vector Tile
- `myschema.mytable` -> **http://localhost:780/myschema.mytable/{z}/{x}/{y}.pbf**

pg_featureserv

- protocol: OGC Api - Features
- format: geojson/json

pg_featureserv: querying features

- filters:
    - bbox: **http://domain.tld/collections/ne.countries/items?bbox=10.4,43.3,26.4,47.7**
    - properties: **http://domain.tld/collections/public.cities/items?continent=Europe&country=France**
- ordering, limits, paging, returned properties etc...
- See OGC Features API

Metadata

- json or web
- includes a preview

pg_tileserv

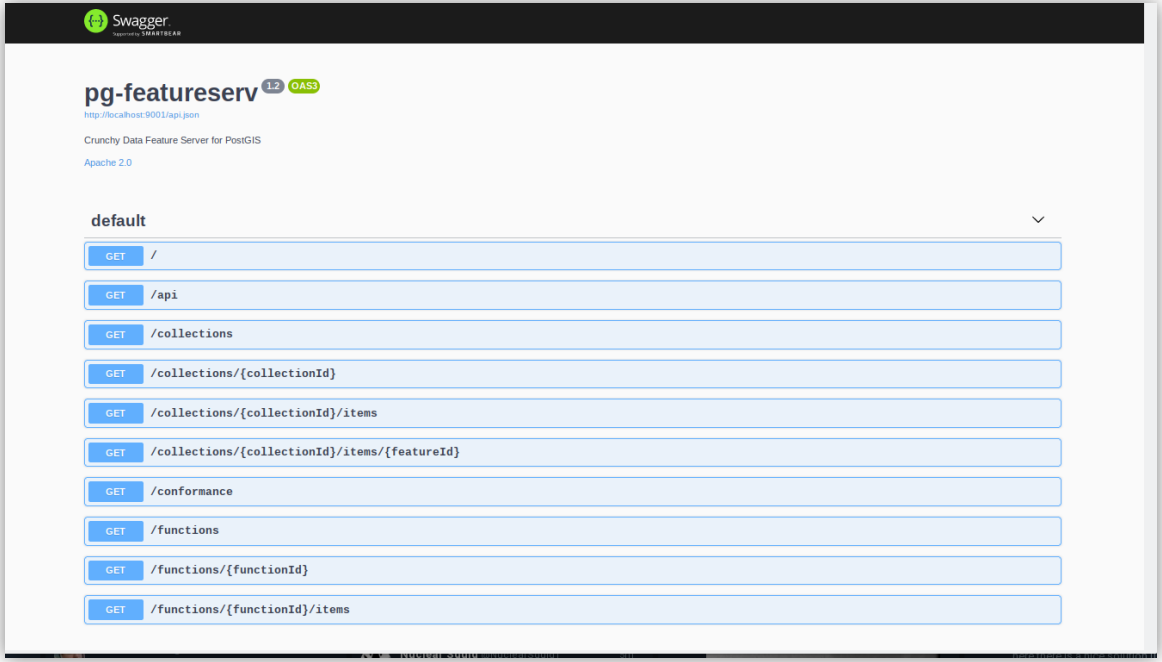
Service Metadata

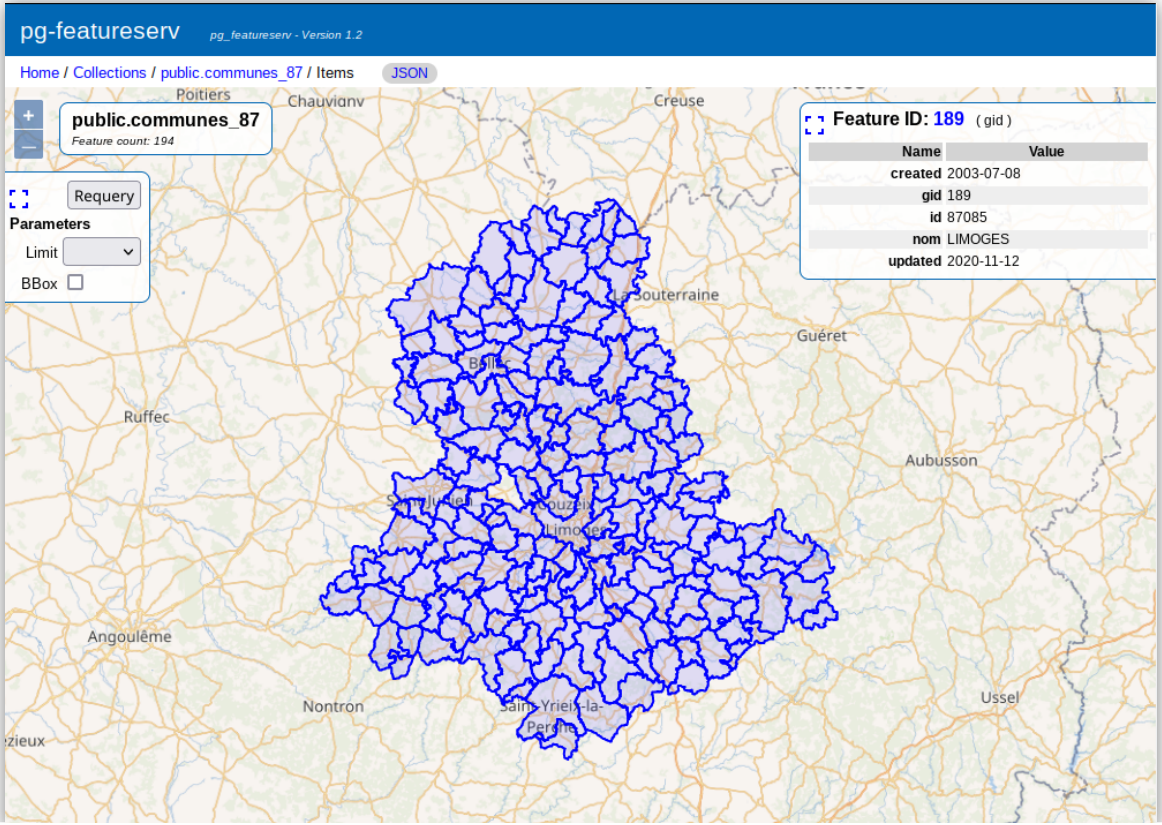
- index.json for layer list

Table Layers

Function Layers

- **visu._debug_tiles** (preview | json)
  Fonction qui permet de visualiser la grille tms (et notamment le z courant) via pg_tileserv
- **visu._sondes_raw** (preview | json)
  Fonction retournant les points directement depuis la table public.sonde. Par défaut, cette fonction empêche tout retour si le nombre de point dépasse 30000, pour éviter de saturer la base. Il est possible de contourner cette limitation avec le paramètre limit_counts -> 'f'
- **visu.sondes** (preview | json)
  Fonction retournant les tuiles de points à partir des coordonnées TMS. Jusqu'au niveau 16 inclus, cette fonction lit la table de cache. Au delà elle lit directement public.sonde


Swagger
pg-featureserv
1.2
OAS3
http://localhost:9001/api.json
Crunchy Data Feature Server for PostGIS
Apache 2.0
default
GET /
GET /api
GET /collections
GET /collections/{collectionId}
GET /collections/{collectionId}/items
GET /collections/{collectionId}/items/{featureId}
GET /conformance
GET /functions
GET /functions/{functionId}
GET /functions/{functionId}/items


pg-featureserv
pg_featureserv - Version 1.2
Home / Collections / public.communes_87 / Items
JSON
public.communes_87
Feature count: 194
Requery
Parameters
Limit
BBox
Feature ID: 189 (gid)
Name
Value
created 2003-07-08
gid 189
id 87085
nom LIMOGES
updated 2020-11-12
Poitiers
Chauvigny
Creuse
La Souterraine
Guéret
Ruffec
Aubusson
Angoulême
Nontron
Ussel
Limoges

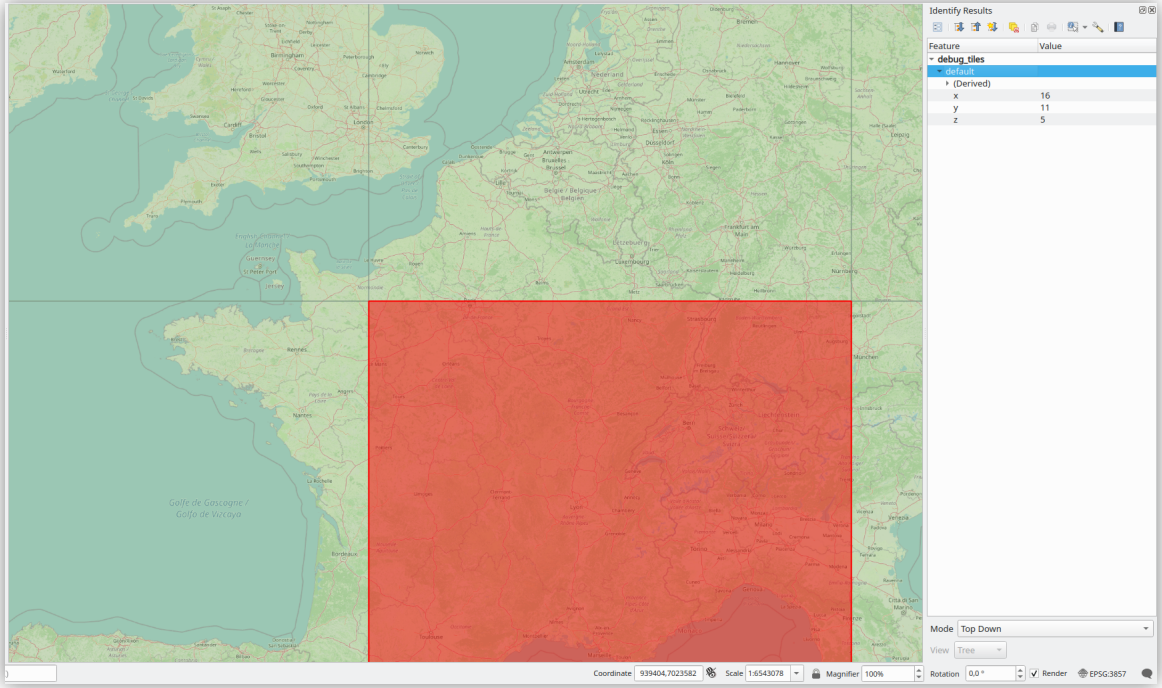

Tulle
2.1564, 46.1569 : 9

functions \o/

```
-- pg_tileserv
create or replace function
visu._debug_tiles(
    z integer, x integer, y integer)
returns bytea
as $$
    with tile as (
        select z, x, y, st_asmvtgeom(

ST_TileEnvelope(z,x,y),

st_TileEnvelope(z,x,y))
    )
    select st_asmvt(tile) from tile;
$$
language 'sql';
```


Identify Results
Feature
Value
debug_tiles
Default
(Derived)
x
16
y
11
z
5
Mode
Top Down
View
Tree
Coordinate
939404,7023582
Scale
1:6543078
Magnifier
100%
Rotation
0,0 °
Render
EPSG:3857

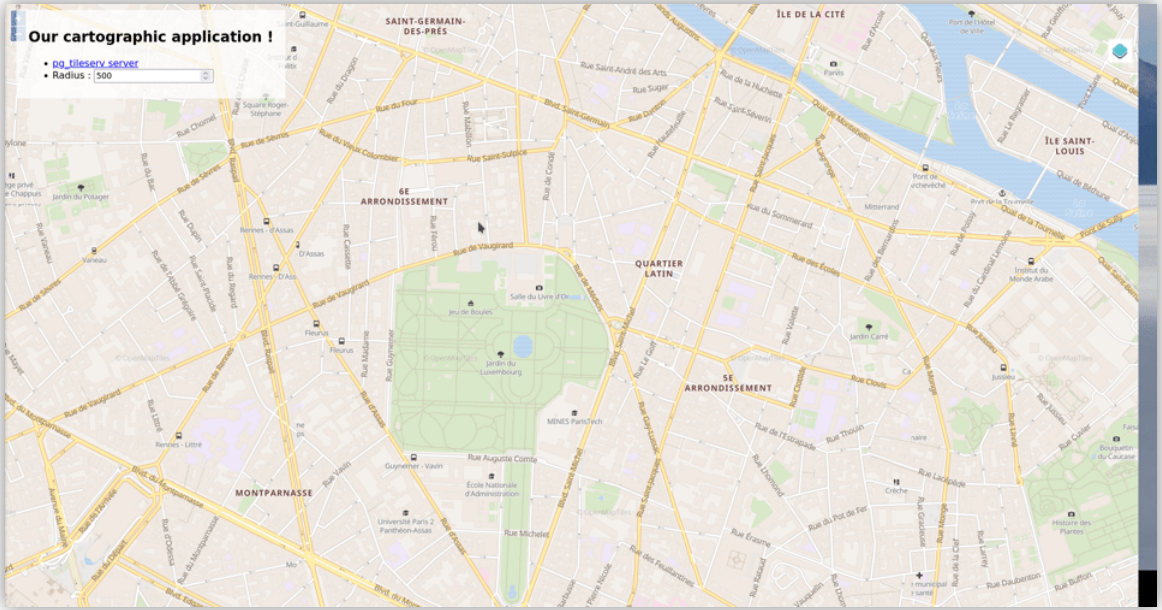
pg_featureserv: functions

```
CREATE OR REPLACE FUNCTION
postgisftw.parcelles_radius(
        click_lon double precision DEFAULT
2.35,
        click_lat double precision DEFAULT
48.86,
        radius double precision DEFAULT
100)
    RETURNS TABLE(
        wkb_geometry geometry,
        id character varying,
        numero character varying,
        contenance numeric
    )
    LANGUAGE sql
    STABLE PARALLEL SAFE
AS $function$
    WITH
```

```
    args AS (
        SELECT
ST_Transform(ST_SetSRID(ST_MakePoint(click
_lon, click_lat), 4326), 2154) AS click
    )
    SELECT
        ST_Transform(
            ST_Intersection(
                p.wkb_geometry,
                ST_Buffer( args.click,
radius)
            ),
            3857
        ) as wkb_geometry,
        p.id,
        p.numero,
        p.contenance
    FROM
        parcelles p
        , args
    WHERE
        ST_DWithin(p.wkb_geometry,
args.click, radius)
```

```
    LIMIT 10000
$function$
```

pg_featureserv: functions



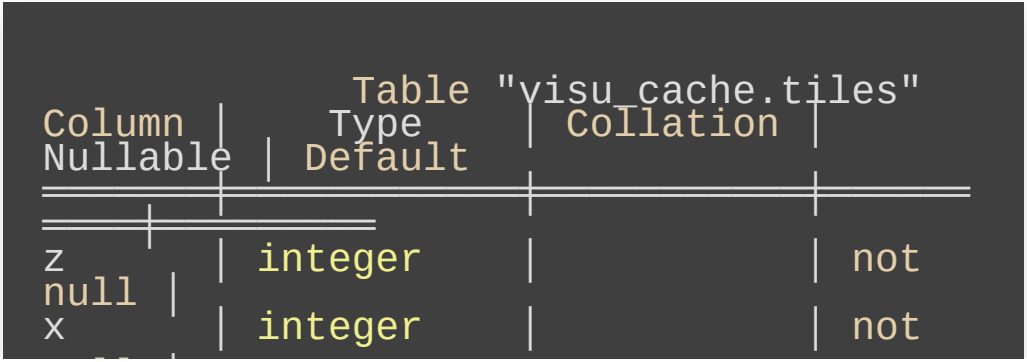
Some functions ideas

- pg_featureserv: setup PostgreSQL's FTS on some properties
- display centers, inscribed circles, aggregation...
- with log table or versioning: get a dataset at a particular time or revision
- pg_tileserv: switch to a cache for low z values
- ... the sky's the limit!

Use case: serving big dataset

Serving pgpointcloud table as mvt:

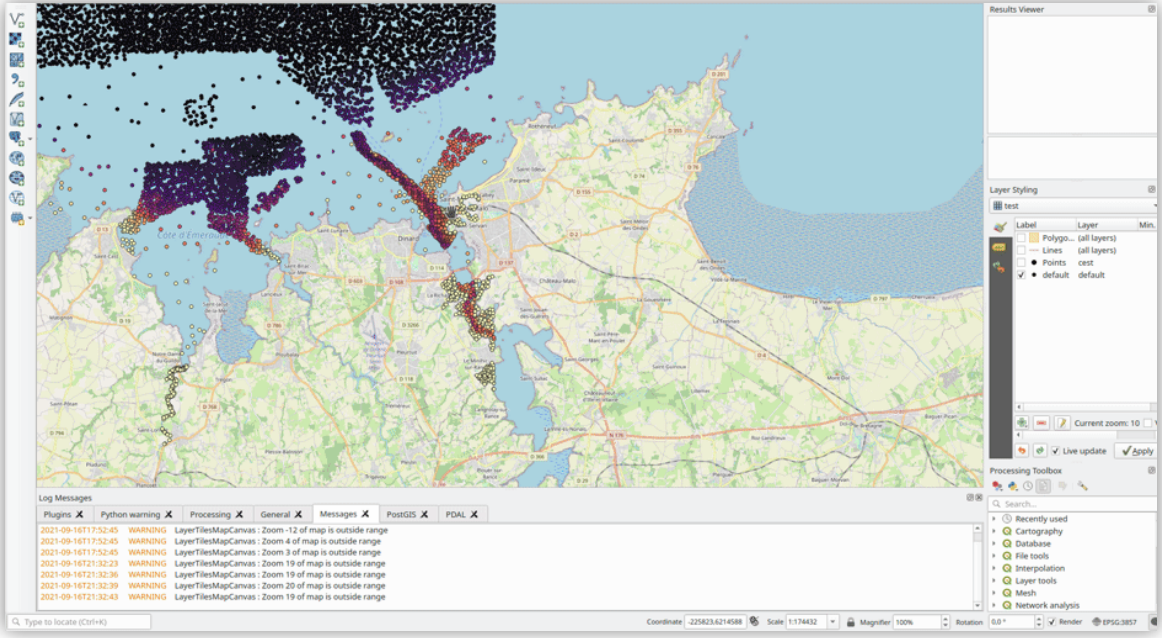
- dataset is big (> 2 billion points, much more planned)
- cache, implemented as a table: z, x, y (indexed) and a patch



```
null |
y      | integer    |           | not
null |
patch  | pcpatch(1) |           | not
null |
```

```
CREATE OR REPLACE FUNCTION visu.sondes(z
integer, x integer, y integer)
 RETURNS bytea
 LANGUAGE plpgsql
 STABLE PARALLEL SAFE SECURITY DEFINER
AS $function$
#variable_conflict use_variable
begin
    if z >= 17 then
        return visu._sondes_raw(z, x, y,
limit_counts => false);
    else
        return (
            with
            mvtgeom as (
                select
                    pc_get(points,
'surfac_id') as surfac_id,
                    pc_get(points, 'z') as
z,
```

```
st_asmvtgeom(st_transform(points::geometry
, 3857), st_tileenvelope(z, x, y)) point
                from
                    visu_cache.tiles,
                    lateral
pc_explode(patch) points
                where
                    tiles.z=z
                    and tiles.x=x
                    and tiles.y=y
            )
            select st_asmvt(mvtgeom) from
mvtgeom
        );
    end if;
end;
$function$
```


Results Viewer
Layer Styling
test
Label
Layer
Min.
Polygo...
(all layers)
Lines
(all layers)
Points
cest
default
default
Current zoom: 10
Live update
Apply
Processing Toolbox
Search...
Recently used
Cartography
Database
File tools
Interpolation
Layer tools
Mesh
Network analysis
Log Messages
Plugins
Python warning
Processing
General
Messages
PostGIS
PDAL
2021-09-16T17:52:45 WARNING LayerTilesMapCanvas : Zoom -12 of map is outside range
2021-09-16T17:52:45 WARNING LayerTilesMapCanvas : Zoom 4 of map is outside range
2021-09-16T17:52:45 WARNING LayerTilesMapCanvas : Zoom 3 of map is outside range
2021-09-16T21:32:23 WARNING LayerTilesMapCanvas : Zoom 19 of map is outside range
2021-09-16T21:32:36 WARNING LayerTilesMapCanvas : Zoom 19 of map is outside range
2021-09-16T21:32:39 WARNING LayerTilesMapCanvas : Zoom 20 of map is outside range
2021-09-16T21:32:43 WARNING LayerTilesMapCanvas : Zoom 19 of map is outside range
Type to locate (Ctrl+K)
Coordinate -225823,6214588
Scale 1:174432
Magnifier 100%
Rotation 0,0 °
Render
EPSG:3857

Pros

- simple
- quality
- flexible (with functions)
- easy to deploy new endpoint, easy to change code

Limitations

- simplicity comes with choice
    - authentication is basic
    - you might still need a backend server
    - you might still need raster layers

Thanks!