Stetl:
Geo-Conversion-Engine for NLExtract and Smart Emission

www.stetl.org

Just van den Broecke
FOSS4GNL 2018
Almere - The Netherlands
July 11, 2018
www.justobjects.nl

About Me

Independent Open Source Geospatial Professional

+ Chair OSGeo Dutch Local Chapter
+ Member of the Dutch OpenGeoGroep

Just van den Broecke
just@justobjects.nl
www.justobjects.nl

OPEN GEO GROEP

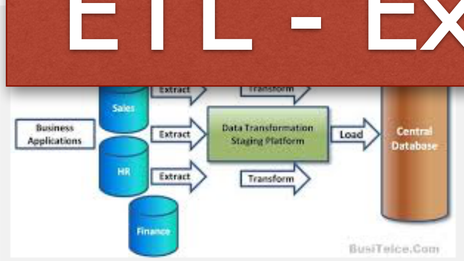
OSGeo·NL

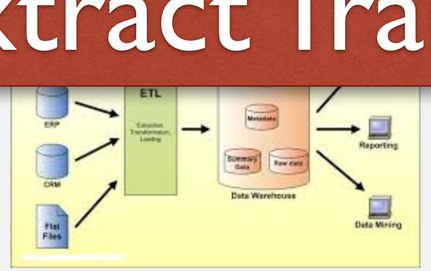
Agenda

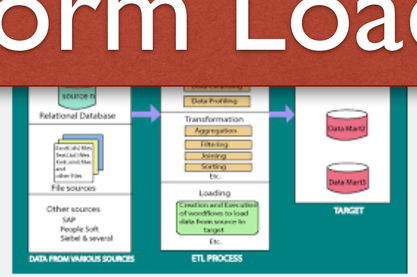
- Spatial ETL
- Stetl Concepts
- **Cases**
  - NLExtract
  - Smart Emission
- Q & A

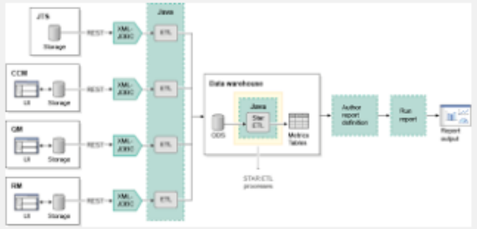
Spatial ETL

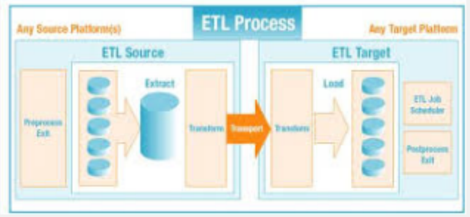
ETL - Extract Transform Load

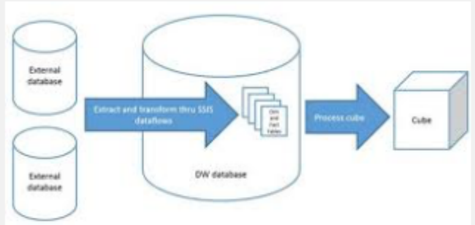


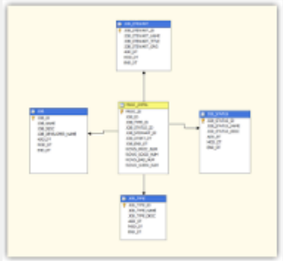


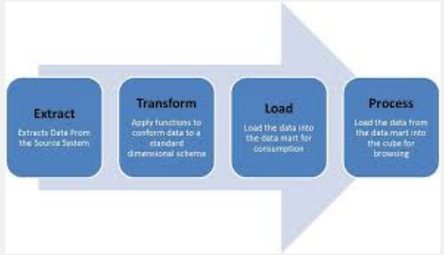


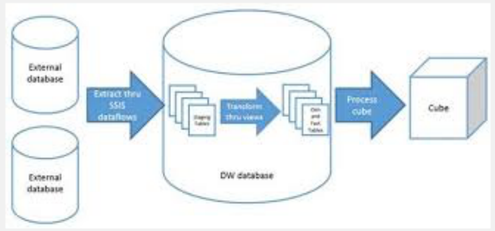


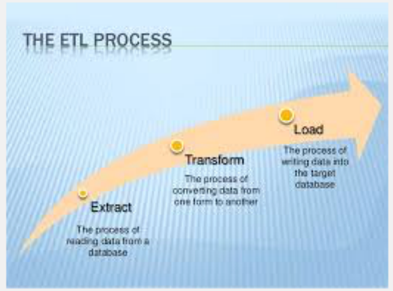


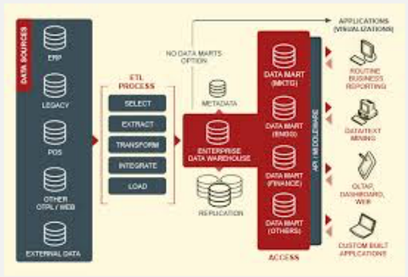


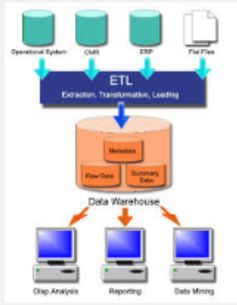


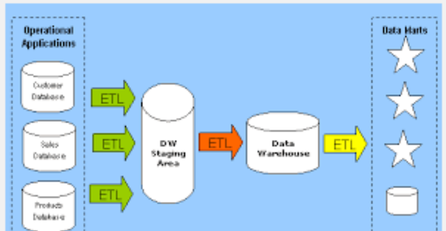


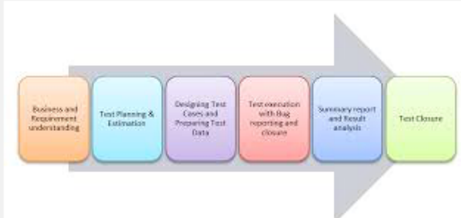


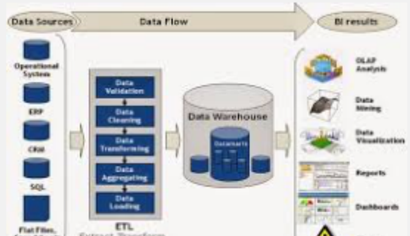




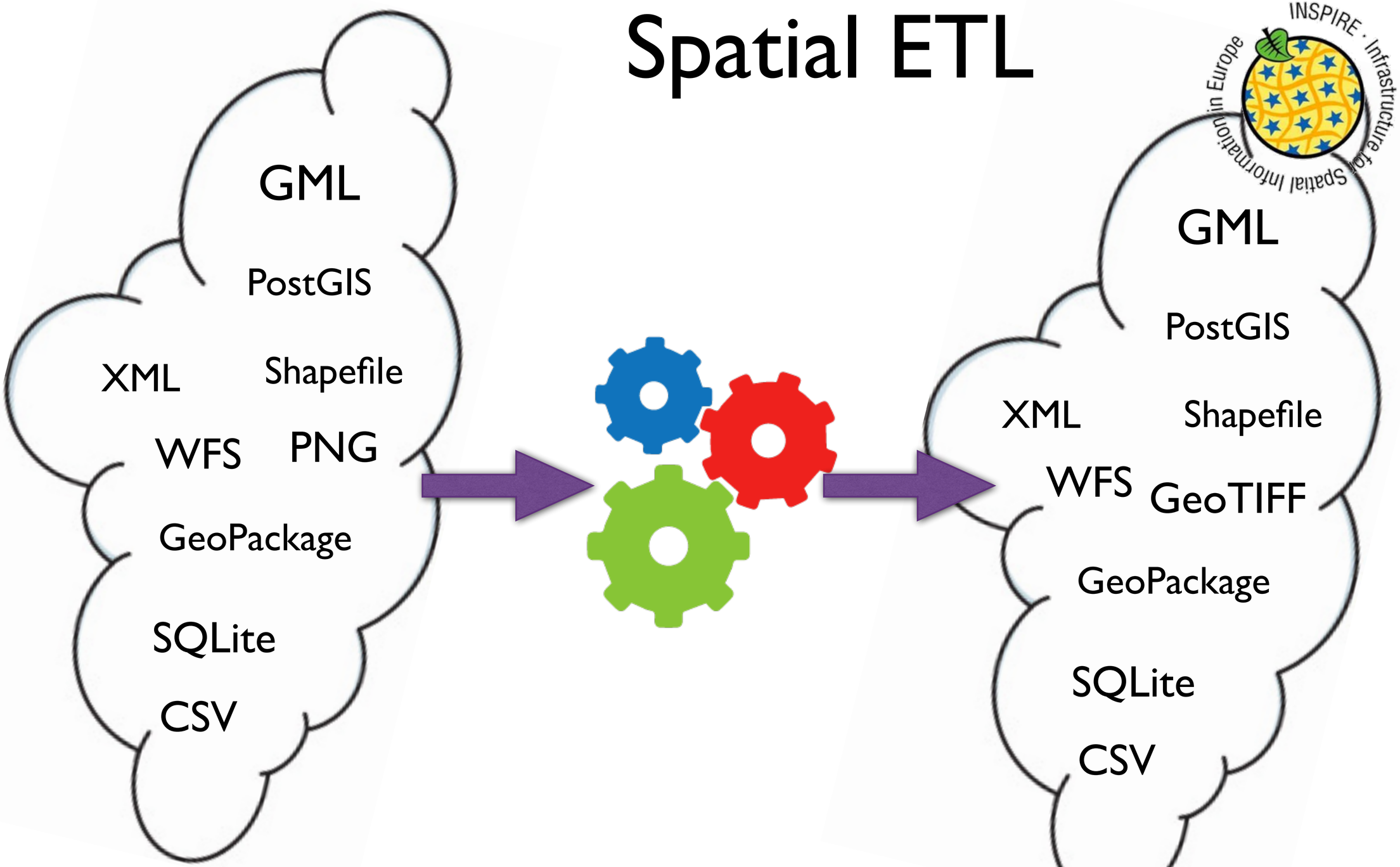








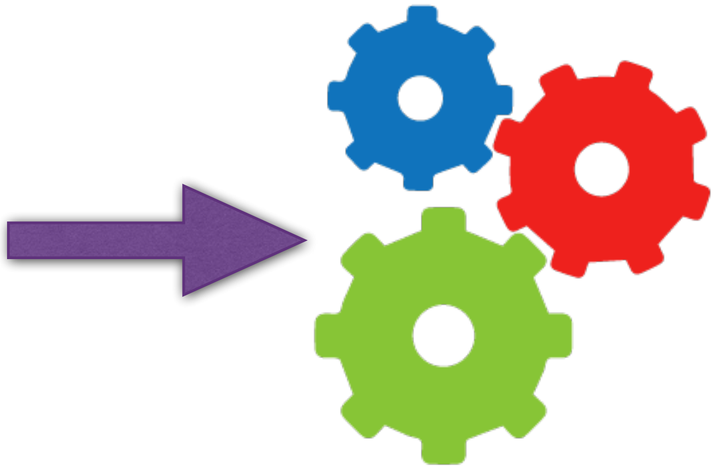
Spatial ETL

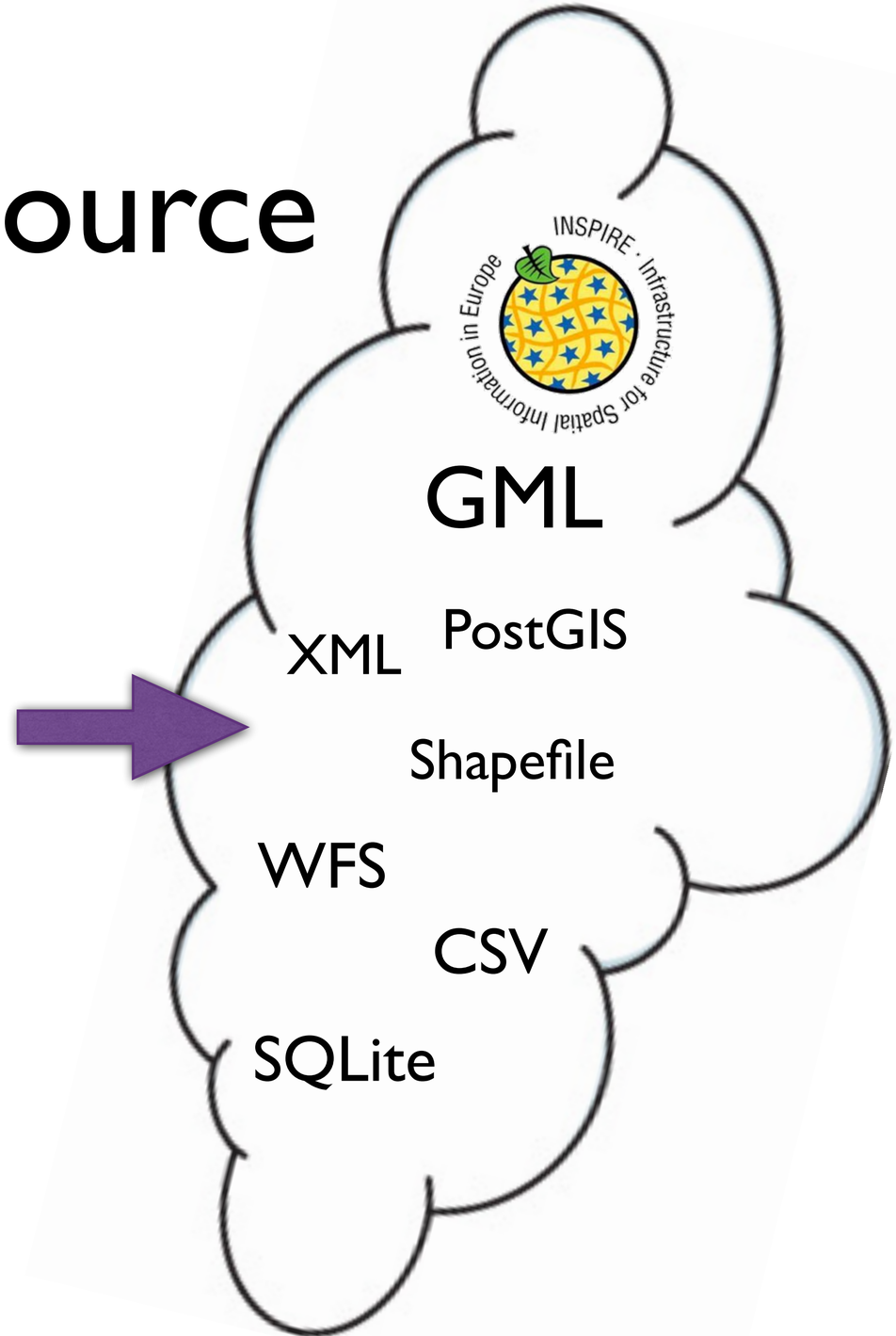

GML
PostGIS
XML
Shapefile
WFS
PNG
GeoPackage
SQLite
CSV
GML
PostGIS
XML
Shapefile
WFS
GeoTIFF
GeoPackage
SQLite
CSV

Spatial ETL

Example non-standard source

```xml
<?xml version='1.0' encoding='utf-8'?>
<cities>
    <city>
        <name>Amsterdam</name>
        <lat>52.4</lat>
        <lon>4.9</lon>
    </city>
    <city>
        <name>Bonn</name>
        <lat>50.7</lat>
        <lon>7.1</lon>
    </city>
    <city>
        <name>Rome</name>
        <lat>41.9</lat>
        <lon>12.5</lon>
    </city>
</cities>
```





GDAL/OGR and PROJ.4

http://www.gdal.org

http://proj.osgeo.org

GDAL for Raster Data - The Geospatial Data Abstraction Library

Arc/Info ASCII Grid, **Arc/Info** Binary Grid (.adf), AIRSAR Polarimetric, Microsoft Windows Device Independent Bitmap (.bmp), BSB Nautical Chart Format (.kap), VTP Binary Terrain Format (.bt), CEOS (Spot for instance), First Generation USGS DOQ (.doq), DODS / OPeNDAP, New Labelled USGS DOQ (.doq), Military Elevation Data (.dt0, .dt1), **ERMapper** Compressed Wavelets (.ecw), **ESRI** .hdr Labelled, ENVI .hdr Labelled Raster, Envisat Image Product (.n1), EOSAT FAST Format, FITS (.fits), Graphics Interchange Format (**.gif**), GMT Compatible netCDF, **GRASS Rasters**, Golden Software ASCII Grid, Golden Software Binary Grid, Golden Software **Surfer** 7 Binary Grid, TIFF / **GeoTIFF** (.tif), GXF - Grid eXchange File, Hierarchical Data Format Release 4 (**HDF4**), Hierarchical Data Format Release 5 (HDF5), **Erdas Imagine** (.img), Vexcel MFF2, **Idrisi** Raster, Image Display and Analysis (WinDisp), ILWIS Raster Map (.mpr,.mpl), Japanese DEM (.mem), **JPEG** JFIF (.jpg), **JPEG2000** (JPEG2000, JP2KAK, JP2ECW, JP2MrSID), NOAA Polar Orbiter Level 1b Data Set (**AVHRR**), Erdas 7.x .LAN and .GIS, Daylon Leveller Heightfield, In Memory Raster, Vexcel MFF, Multi-resolution Seamless Image Database, Meteosat Second Generation, NDF, NITF, **NetCDF**, OGDI Bridge, PCI .aux Labelled, PCI Geomatics Database File, Portable Network Graphics (**.png**), PCRaster (.map), Netpbm (.ppm,.pgm), Swedish Grid RIK (.rik), RadarSat2 XML (product.xml), **ArcSDE** Raster, USGS SDTS DEM (*CATD.DDF), Raster Matrix Format (*.rsw, .mtw), SAR CEOS, **SGI** Image Format, USGS ASCII DEM (.dem), OGC Web Coverage Server, X11 Pixmap (.xpm)

OGR for Vector Data - Simple Feature Library

Arc/Info Binary Coverage, Comma Separated Value (**.csv**), DODS/OPeNDAP, DWG, **DXF**, ESRI Personal GeoDatabase, ESRI ArcSDE, **ESRI Shapefile**, FMEObjects Gateway, GML, **GMT Mapping**, **GRASS Vectors**, INTERLIS, **Google Earth KML**, Mapinfo File, Microstation DGN, Spatial **MySQL**, OGDI Vectors, ODBC generic database access layer, **Oracle** Spatial, PostgreSQL **PostGIS**, S-57 (ENC), SDTS, SQLite, UK .NTF, U.S. Census TIGER/Line, VRT - Virtual Datasource, Informix DataBlade

From: https://live.osgeo.org/en/overview/gdal_overview.html

Kersten
@Fernerkundung

Follow

When people describe #GDAL als the "swiss army knife of GIS data" this is what I think of.

RETWEETS 20 LIKES 21

12:49 PM - 25 Jan 2016

20 21

Plenty of Tools…

ogr2ogr

Each tool is powerful by itself but cannot do the entire ETL

FOSS ETL - How to Combine Components?

+

+

+ .. = ?

Solution: Add Python to the Equation

(+ + + ..) =

Stetl

Stetl

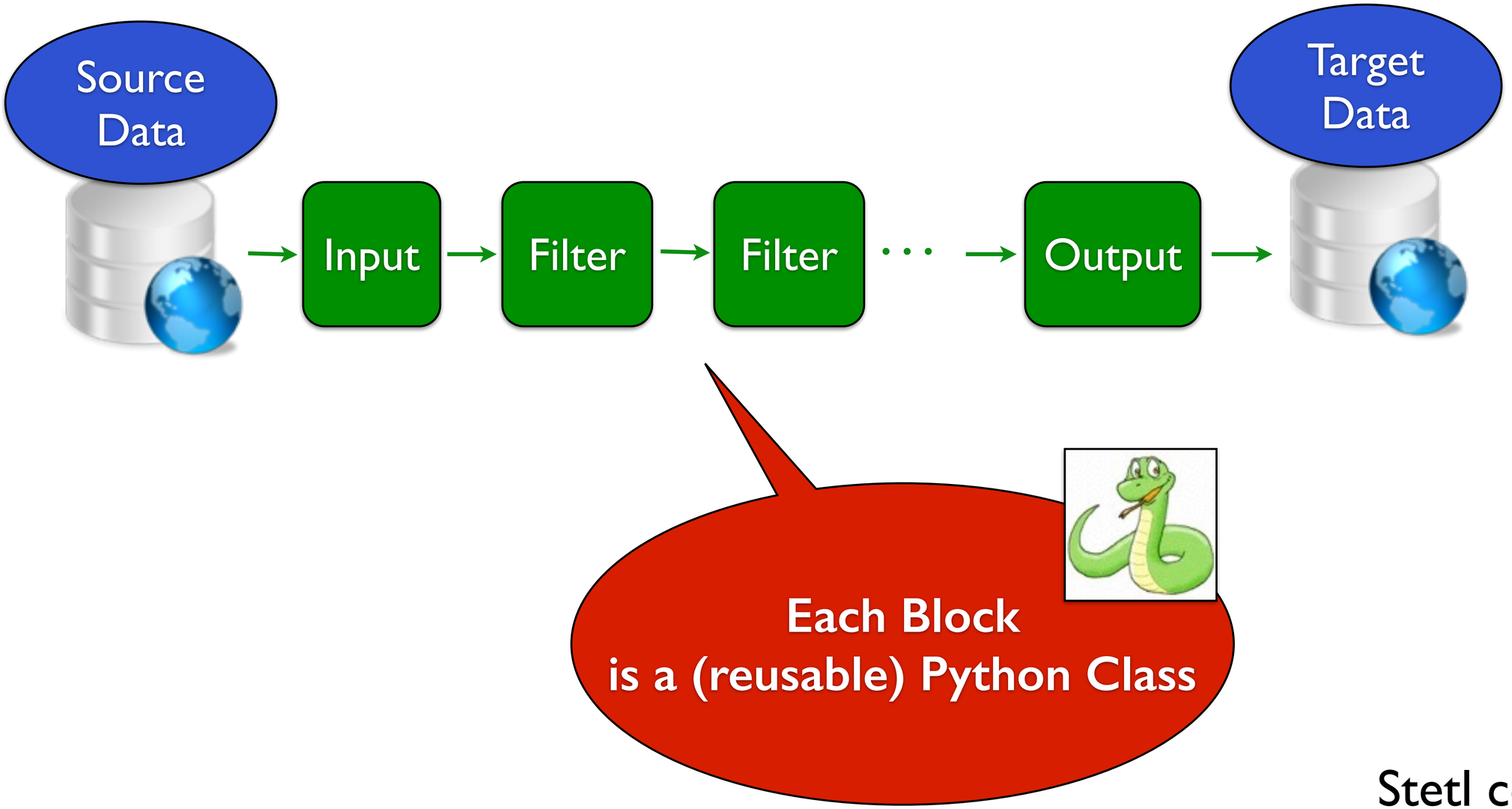
=

Simple
Streaming
Spatial
Speedy
ETL

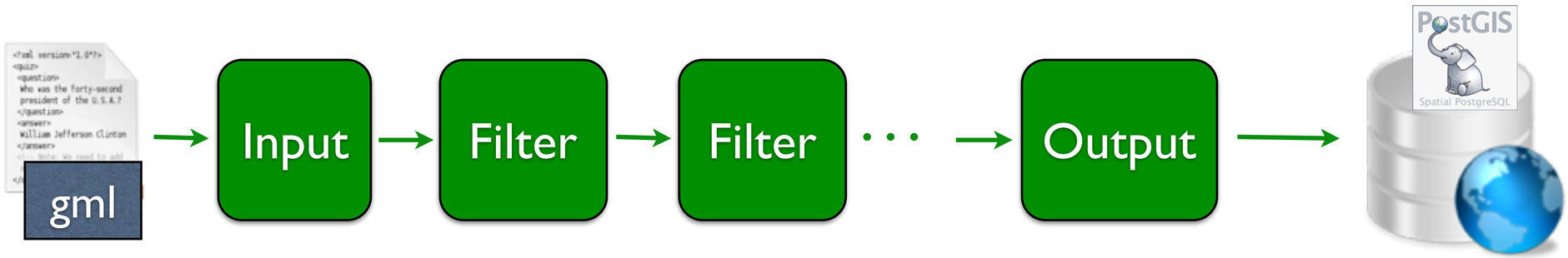
And what about?

Stetl
Concepts

Process Chain

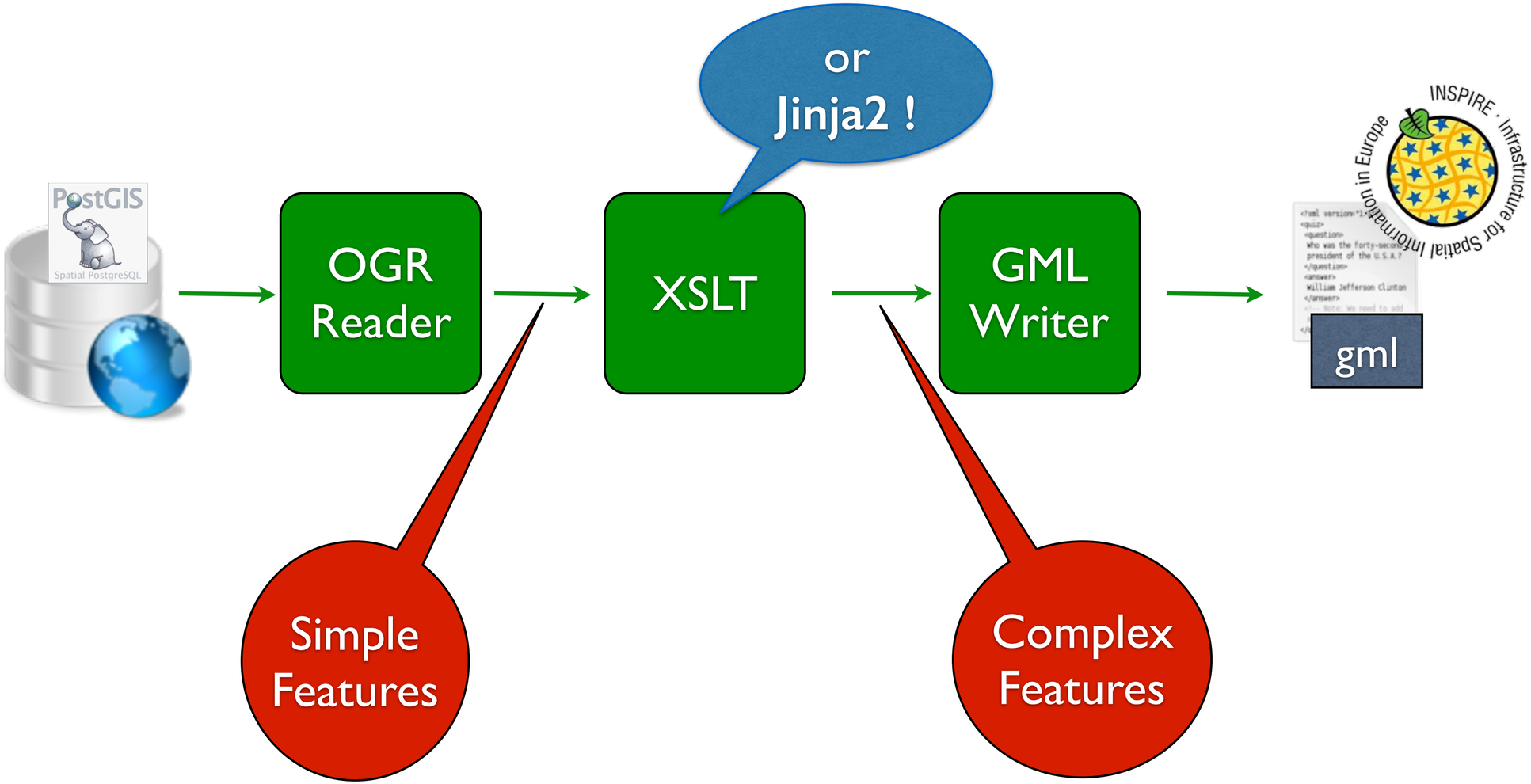


Process Chain - GML to PostGIS

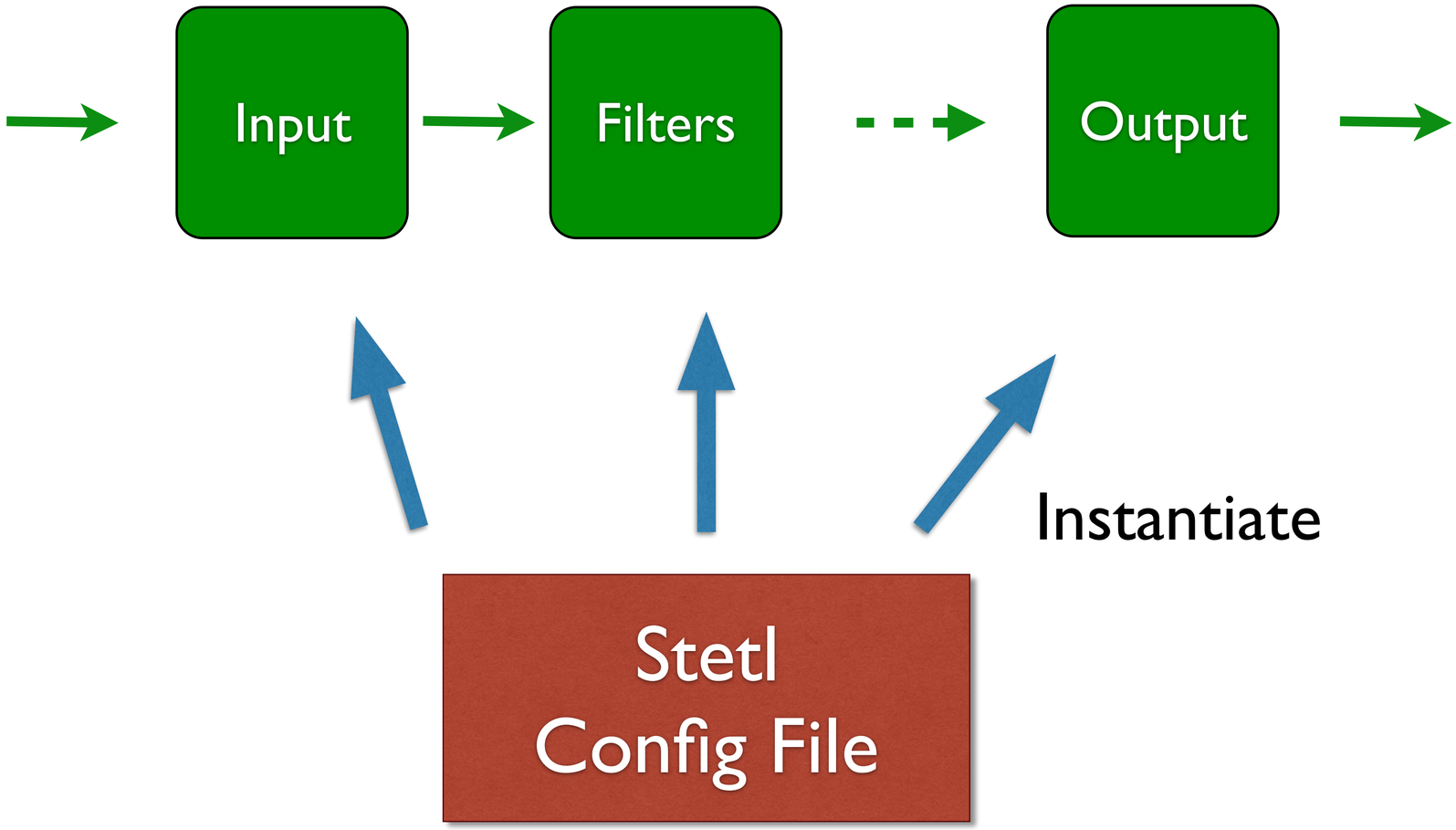


Stetl concepts

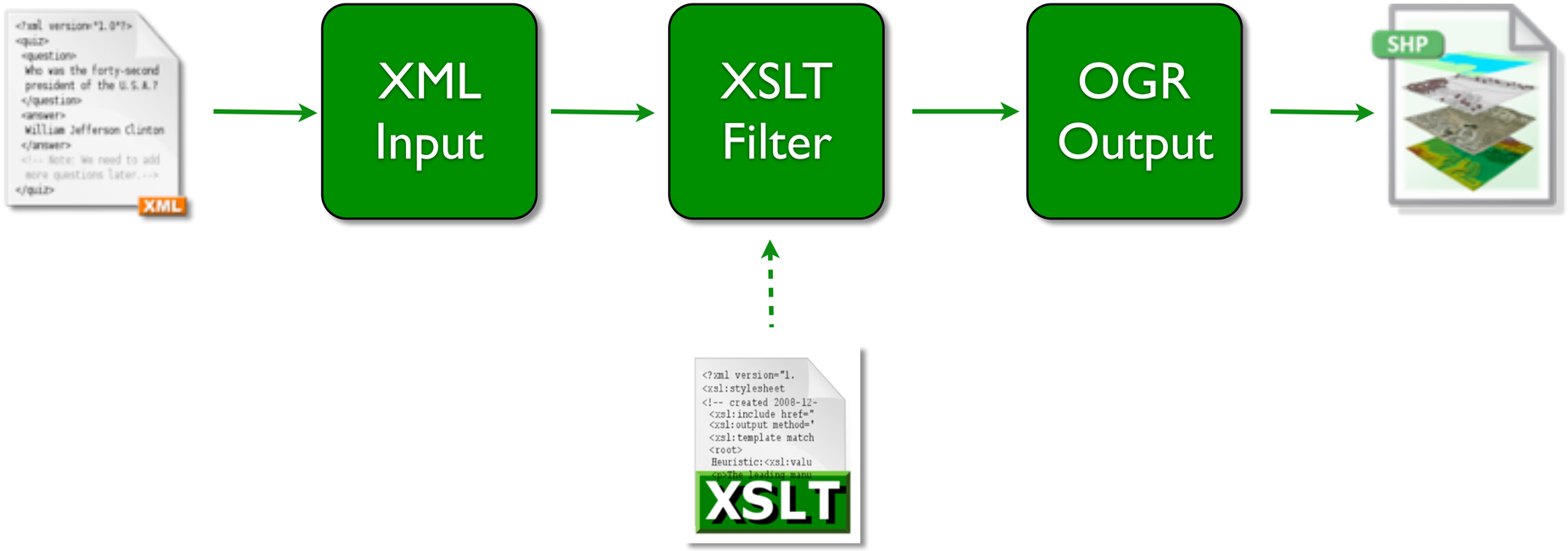
Example: Data Model Transform



Process Chain - How?



Example: XML File to Shapefile

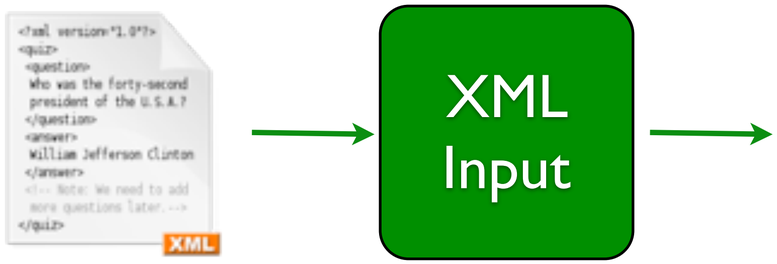


Example: XML to Shape

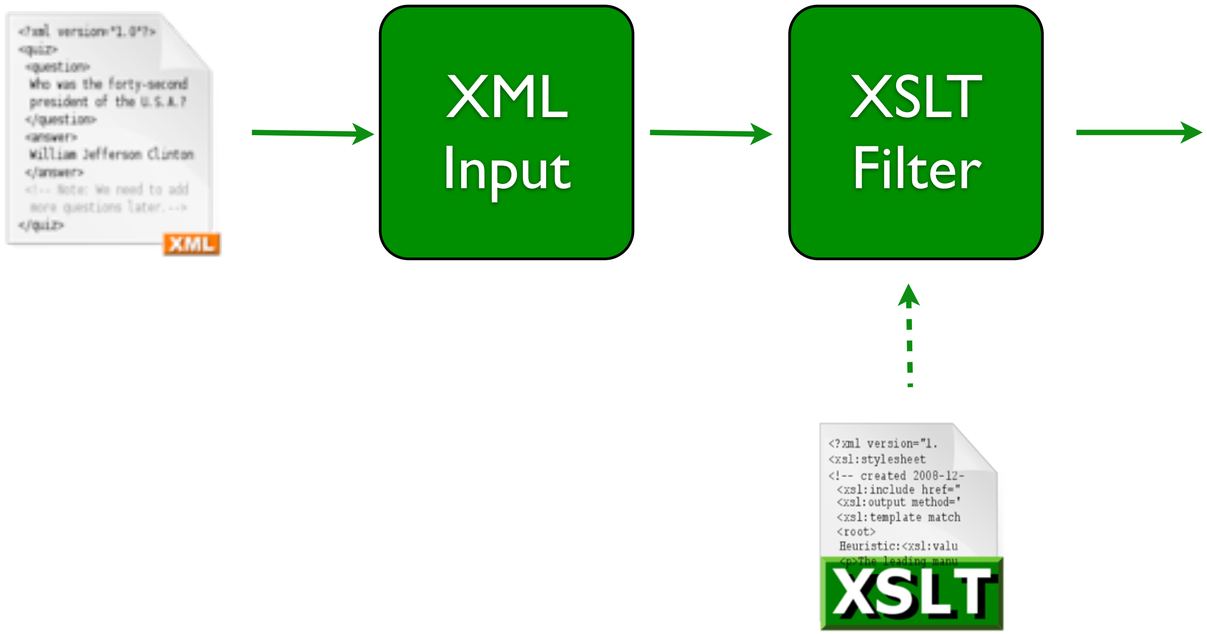
```xml
<?xml version='1.0' encoding='utf-8'?>
<cities>
    <city>
        <name>Amsterdam</name>
        <lat>52.4</lat>
        <lon>4.9</lon>
    </city>
    <city>
        <name>Bonn</name>
        <lat>50.7</lat>
        <lon>7.1</lon>
    </city>
    <city>
        <name>Rome</name>
        <lat>41.9</lat>
        <lon>12.5</lon>
    </city>
</cities>
```

The Source File

Example: XML to Shape



Example: XML to Shape



Example: XML to Shape

```
    <xsl:template match="/">
        <ogr:FeatureCollection
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ogr="http://ogr.maptools.org/"
                xmlns:gml="http://www.opengis.net/gml"
                xsi:schemaLocation="http://ogr.maptools.org/ ../gmlcities.xsd  http://www.opengis.net/gml http://schemas
                >
            <gml:boundedBy>
              <gml:Box>
                <gml:coord><gml:X>-180.0</gml:X><gml:Y>-90.0</gml:Y></gml:coord>
                <gml:coord><gml:X>180.0</gml:X><gml:Y>90.0</gml:Y></gml:coord>
              </gml:Box>
            </gml:boundedBy>
             <!-- Loop through all cities. -->
            <xsl:apply-templates/>
        </ogr:FeatureCollection>
    </xsl:template>

    <!-- Make each city an ogr:featureMember. -->
    <xsl:template match="city">
        <gml:featureMember>
            <ogr:City>
                <ogr:name>
                    <xsl:value-of select="name"/>
                </ogr:name>
                <ogr:geometry>
                    <gml:Point srsName="urn:ogc:def:crs:EPSG:4326">
                        <gml:coordinates><xsl:value-of select="lat"/>,<xsl:value-of select="lon"/></gml:coordinates>
                     </gml:Point>
                </ogr:geometry>
            </ogr:City>
        </gml:featureMember>
    </xsl:template>
</xsl:stylesheet>
```

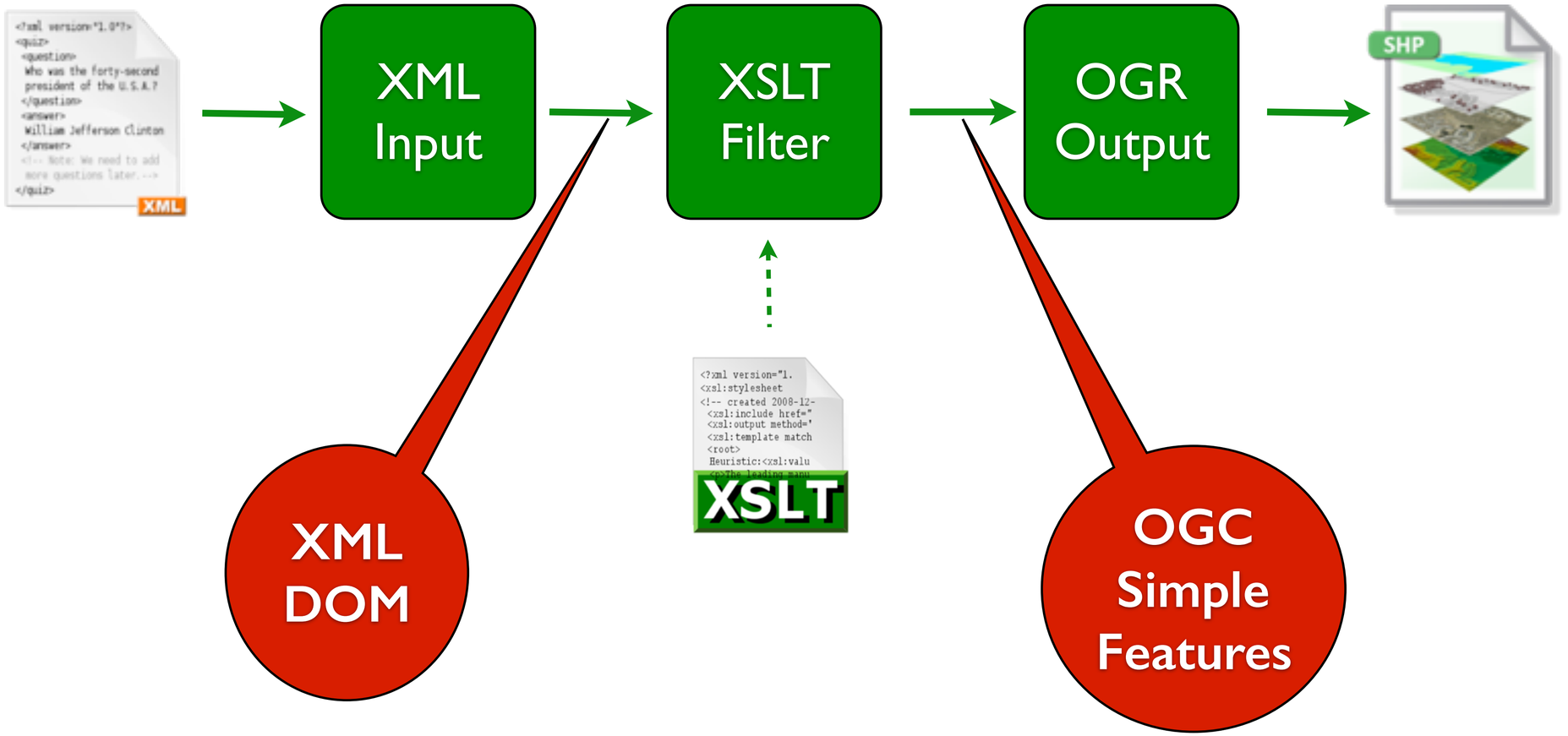
Prepare XSLT Script

Example: XML to Shape

```
<?xml version='1.0' encoding='utf-8'?>
<ogr:FeatureCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogr="http://
  <gml:boundedBy>
    <gml:Box>
      <gml:coord>
        <gml:X>-180.0</gml:X>
        <gml:Y>-90.0</gml:Y>
      </gml:coord>
      <gml:coord>
        <gml:X>180.0</gml:X>
        <gml:Y>90.0</gml:Y>
      </gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ogr:City>
      <ogr:name>Amsterdam</ogr:name>
      <ogr:geometry>
        <gml:Point srsName="urn:ogc:def:crs:EPSG:4326">
          <gml:coordinates>52.4,4.9</gml:coordinates>
        </gml:Point>
      </ogr:geometry>
    </ogr:City>
  </gml:featureMember>
  <gml:featureMember>
    <ogr:City>
      <ogr:name>Bonn</ogr:name>
      <ogr:geometry>
        <gml:Point srsName="urn:ogc:def:crs:EPSG:4326">
          <gml:coordinates>50.7,7.1</gml:coordinates>
        </gml:Point>
      </ogr:geometry>
    </ogr:City>
  </gml:featureMember>
```

XSLT GML Output

Example: XML to Shape



Example: XML to Shape

```
# Transform input xml to valid GML file and then to Shape.

[etl]
chains = input_xml_file | transformer_xslt | output_ogr_shape

[input_xml_file]
class = inputs.fileinput.XmlFileInput
file_path = input/cities.xml

[transformer_xslt]
class = filters.xsltfilter.XsltFilter
script = cities2gml.xsl

# The ogr2ogr command-line.
[output_ogr_shape]
class = outputs.ogroutput.Ogr2OgrOutput
temp_file = temp/gmlcities.gml
ogr2ogr_cmd = ogr2ogr \
    -overwrite \
    -f "ESRI Shapefile" \
    -a_srs epsg:4326 \
   output/gmlcities.shp \
   temp/gmlcities.gml
```

Process Chain

XML Input

XSLT Filter

ogr2ogr Output

The Stetl Config File

Data Structures

- Components exchange *Packets*
- *Packet* contains data and status
- Data formats, e.g. :

  xml_line_stream
  etree_doc
  etree_element (feature)
  etree_element_array
  record (dict)
  string
  any
  .
  .

Running Stetl

```
stetl -c etl.cfg
```

```
stetl -c etl.cfg [-a <properties>]
```

Example Components

XMLFile	XSLT	GML
ogr2ogr	XMLAssembler	GDAL/OGR
LineStream	XMLValidator	WFS-T
Postgres/PostGIS	Jinja2	Postgres/PostGIS
CSV, JSON etc	FeatureExtractor	SOS, SensorThings API
YourInput	*YourFilter*	*YourOutput*

Example: XsltFilter Python Code

```
class XsltFilter(Filter):
    """
    Invokes XSLT processor (via lxml) for given XSLT script on an etree doc.
    """

    @Config(ptype=str, required=True)
    def script(self):
        """
        Path to XSLT script file.
        """
        pass

    def __init__(self, configdict, section):
        Filter.__init__(self, configdict, section, consumes=FORMAT.etree_doc, produces=FORMAT.etree_doc)

        self.xslt_file = open(self.script, 'r')

        # Parse XSLT file only once
        self.xslt_doc = etree.parse(self.xslt_file)
        self.xslt_obj = etree.XSLT(self.xslt_doc)
        self.xslt_file.close()

    def invoke(self, packet):
        packet.data = self.xslt_obj(packet.data)
        return packet
```

Your Own Components

```
class MyFilter(Filter):
    # Constructor
    def __init__(self, configdict, section):
        Filter.__init__(self, configdict, section, consumes=FORMAT.etree_doc,
                                                   produces=FORMAT.etree_doc)

    def invoke(self, packet):
        log.info("CALLING MyFilter OK!!!!")
        return packet
```

Step 1- Define Class

```
[etl]
chains = input_xml_file|my_filter|output_std

[input_xml_file]
class = inputs.fileinput.XmlFileInput
file_path = input/cities.xml

# My custom component
[my_filter]
class = my.myfilter.MyFilter

[output_std]
class = outputs.standardoutput.StandardXmlOutput
```

Step 2- Configure Class

Installing Stetl - PyPi

```
sudo pip install stetl
```

Deps

- GDAL+Python bindings
- lxml (xml proc)
- psycopg2 (Postgres)

Installing Stetl - Docker

https://hub.docker.com/r/geopython/stetl

Case 1 - NLExtract

What is NLExtract?

ETL for Dutch National Open Datasets

http://nlextract.nl

- GML to PostGIS:
  BAG, BRT (Top10NL), BGT, BRK (Cadastral Parcels)
- Downloads data.nlextract.nl

app.map5.nl

Otterlo

N304

APELDOORNSEWG

ARNHEMSEWG

POTHOVENLN

BOUMANLN

DORPSSTR

SPORTLN

ZANDINGSWG

ONDERLANGS

HOUTKAMPSWG

EDESEWG

ELZEBOSSERPAD

HAARWG

KOEWG

MOSSELSEPAD

MOSSELSEWG

BUURTWG

BUURTBOSLN

EGYPTE

DE DENNEN

BOSPAD

CYPRESLN

HEIDERAND

LARIXPAD

SPARRELN

Grand Cafe 't Kruller

't Witte Hoes / Recept 35

Nederlands Tegelmuseum

Canergie cottage

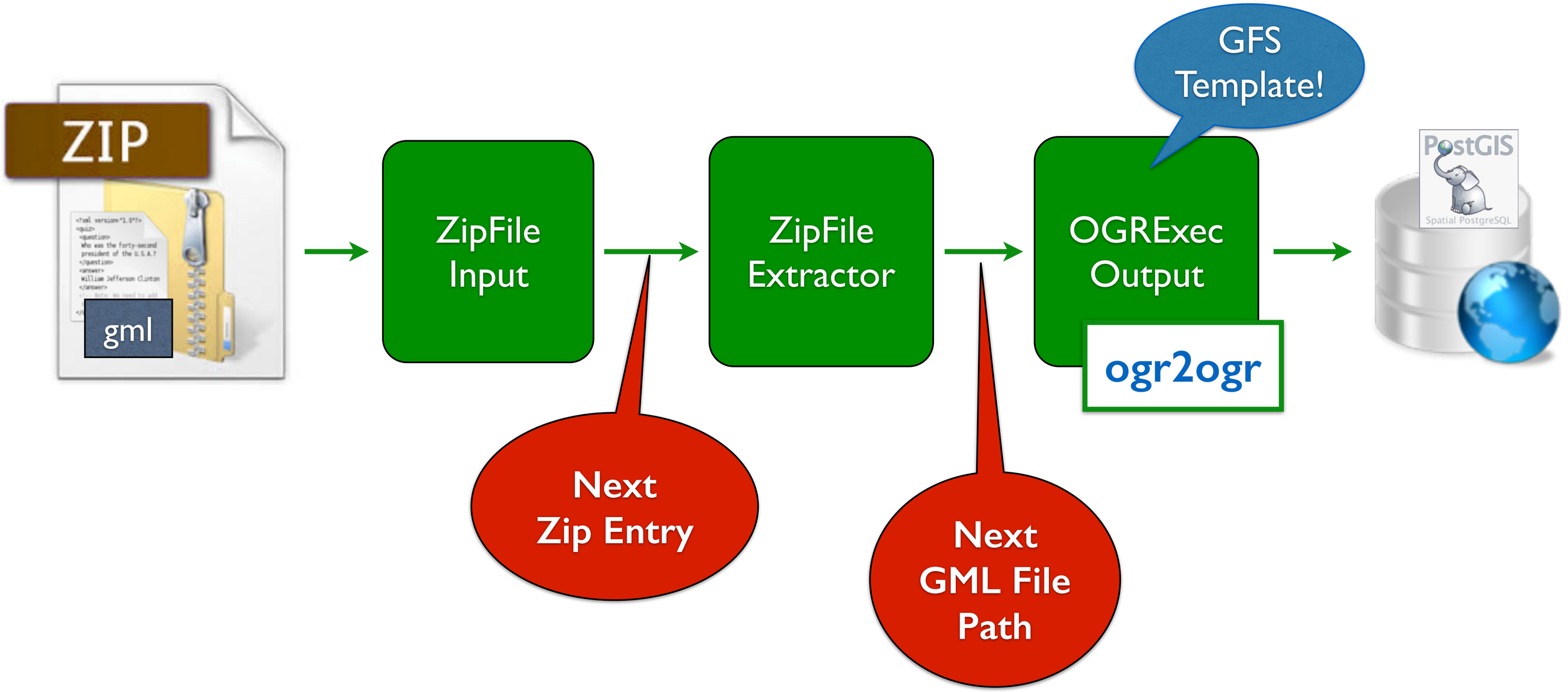
't Rakkertje

BRK - Input GML from PDOK

```
$ ls -lh
total 3.6G
-rw-rw-r-- 1 mlx mlx 3.6G May  7 10:32 kadastralekaartv3-gml-nl-nohist.zip

$ unzip -l kadastralekaartv3-gml-nl-nohist.zip

 Length      Date    Time    Name
---------  ---------- -----   ----
16562827212  2018-01-01 13:42   Bebouwing.gml
 4737201554  2018-01-02 09:00   Annotatie.gml
20883080403  2018-05-01 15:15   Perceel.gml
27532931147  2018-05-01 20:19   Kadastralegrens.gml
---------                     -------
69716040316                    4 files
```

About 70GB in 4 GML Files!

Stetl Config BRK to PostGIS



Stetl Config - BRK .zip to PostGIS

```
[etl]
chains = input_zip_file|extract_zip_file|output_ogr2ogr

[input_zip_file]
class=inputs.fileinput.ZipFileInput
file_path = {input_dir}
filename_pattern = {zip_files_pattern}
name_filter = {filename_match}

# Filter to extract a ZIP file one by one to a temporary location
[extract_zip_file]
class=filters.zipfileextractor.ZipFileExtractor
file_path = {temp_dir}/fromzip-tmp.gml

[output_ogr2ogr]
class = outputs.execoutput.Ogr2OgrExecOutput
# destination format: OGR vector format name
dest_format = PostgreSQL
# destination datasource: name of datasource
dest_data_source = "PG:dbname={database} host={host} port={port} user={user} password={password} active_schema={schema}"
# layer creation options will only be added to ogr2ogr on first run
lco = -lco LAUNDER=YES -lco PRECISION=NO
# spatial_extent, translates to -spat xmin ymin xmax ymax
spatial_extent = {spatial_extent}
# gfs template
gfs_template = {gfs_template}
# miscellaneous ogr2ogr options
options = -append -gt 65536 {multi_opts} --config PG_USE_COPY NO --config CPL_ZIP_ENCODING CP437
```

GFS for Table+Column Mapping

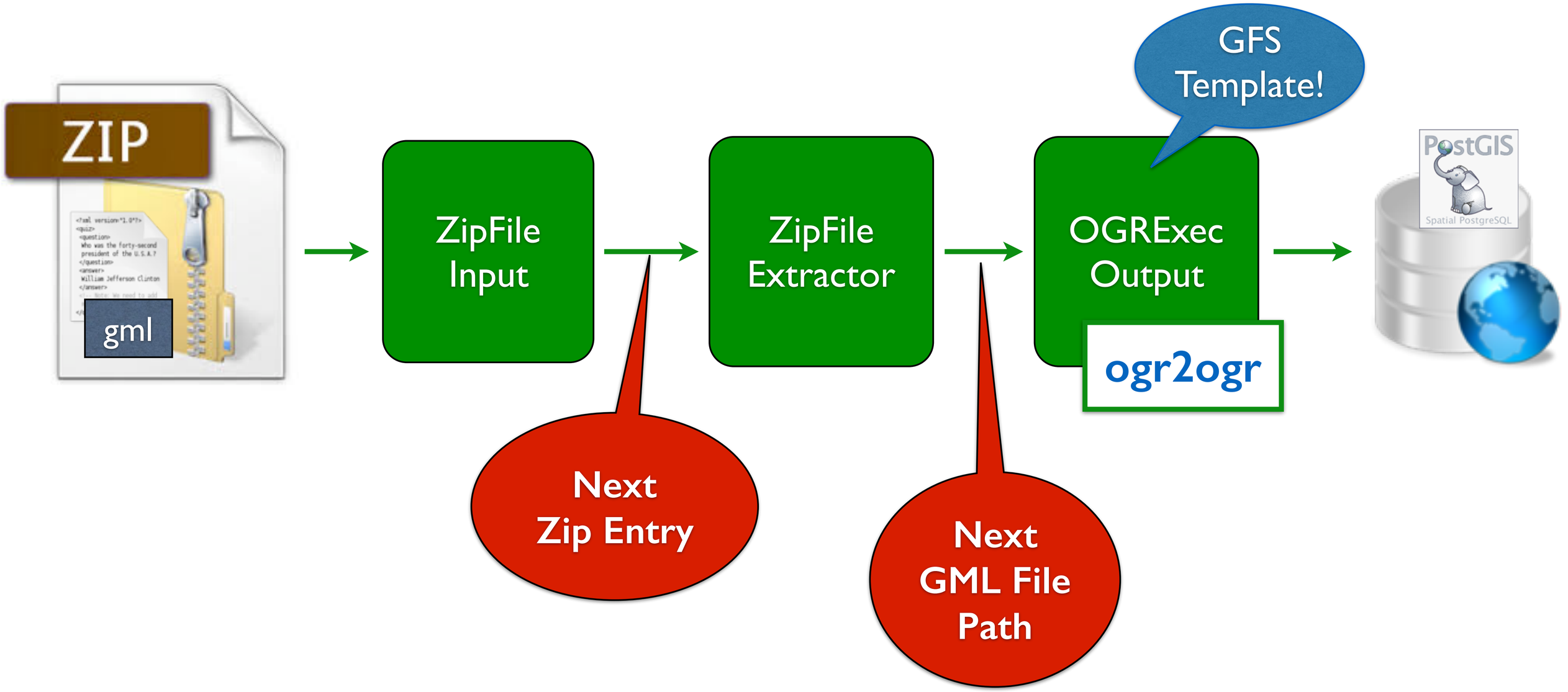
```
<GMLFeatureClassList>
    .
    <GMLFeatureClass>
        <Name>Perceel</Name>
        <ElementPath>Perceel</ElementPath>
        <!--<GeometryType>1</GeometryType>-->
        <SRSName>urn:ogc:def:crs:EPSG::28992</SRSName>
         .
        <PropertyDefn>
            <Name>gemeente</Name>
            <ElementPath>kadastraleAanduiding|TypeKadastraleAanduiding|kadastraleGemeente|KadastraleGemeenteKeuze|
AKRKadastraleGemeenteCode</ElementPath>
            <Type>String</Type>
            <Width>5</Width>
        </PropertyDefn>
        <PropertyDefn>
            <Name>sectie</Name>
            <ElementPath>kadastraleAanduiding|TypeKadastraleAanduiding|sectie</ElementPath>
            <Type>String</Type>
            <Width>2</Width>
        </PropertyDefn>
        <PropertyDefn>
            <Name>perceelnummer</Name>
            <ElementPath>kadastraleAanduiding|TypeKadastraleAanduiding|perceelnummer</ElementPath>
            <Type>Integer</Type>
        </PropertyDefn>
        .
        <GeomPropertyDefn>
            <Name>begrenzing</Name>
            <!-- OGR geometry name -->
            <ElementPath>begrenzingPerceel</ElementPath>
            <!-- XML element name possibly with '|' to specify the path -->
            <Type>Polygon</Type>
        </GeomPropertyDefn>
   </GMLFeatureClass>
</GMLFeatureClassList>
```

Resultaat in PostGIS

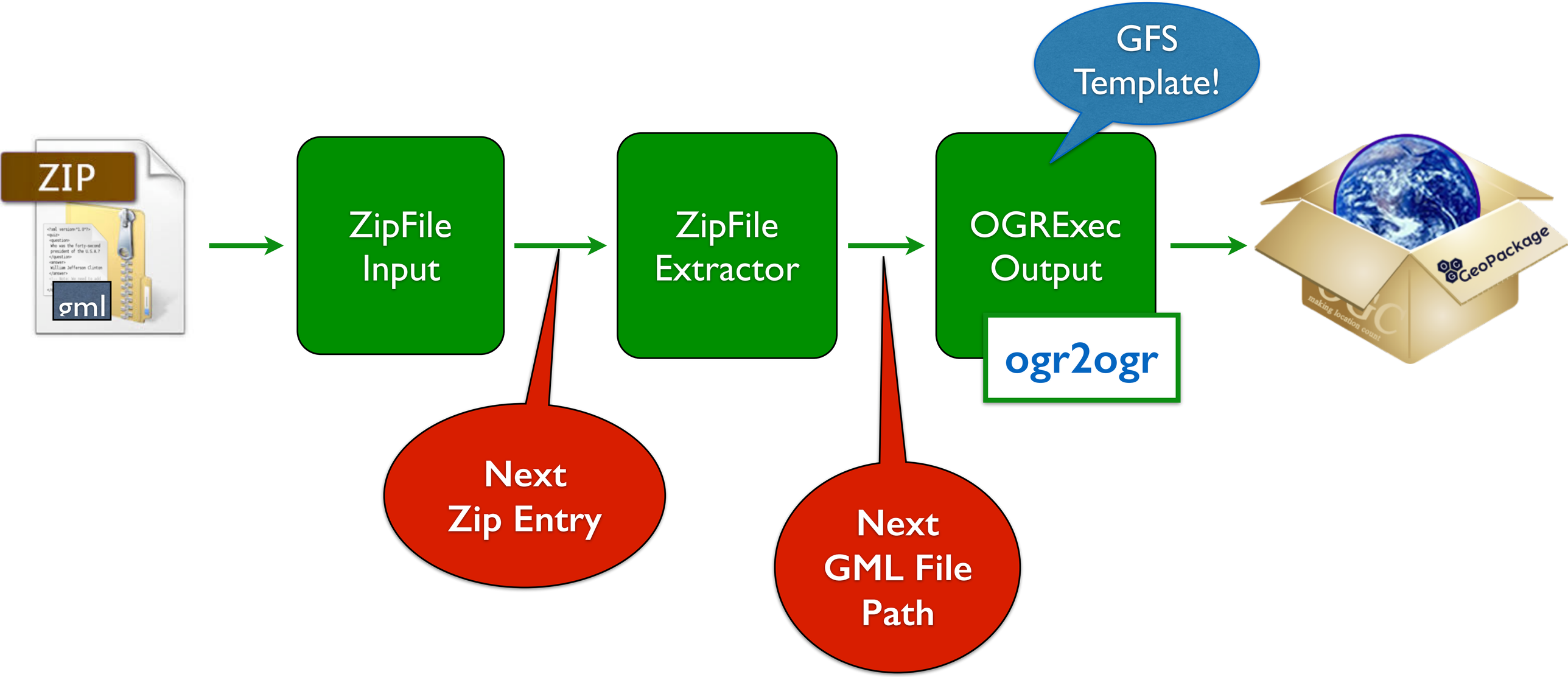
ogc_fid	gml_id	namespace	lokaalid	logischtijdstipontstaan	gemeente	sectie	perceelnummer	waarde
1	NL.IMKAD.KadastraalObject.120031707	NL.IMKAD.KadastraalObject	120031707	2009-03-27T23:59:59.000	AMR04	O	3322	178
2	NL.IMKAD.KadastraalObject.120118807	NL.IMKAD.KadastraalObject	120118807	2009-03-27T23:59:59.000	AMR04	P	3907	446
3	NL.IMKAD.KadastraalObject.120077624	NL.IMKAD.KadastraalObject	120077624	2009-03-27T23:59:59.000	AMR04	O	4482	298
4	NL.IMKAD.KadastraalObject.120119310	NL.IMKAD.KadastraalObject	120119310	2009-03-27T23:59:59.000	AMR04	Q	5520	155
5	NL.IMKAD.KadastraalObject.120095784	NL.IMKAD.KadastraalObject	120095784	2009-03-27T23:59:59.000	AMR04	Q	4128	250

perceelnummerrotatie	deltax	deltay	begrenzing
0	0	0	0103000020407100000100000000C0000003D0AD7A3CF5E014...
0	0	0	0103000020407100000100000006000000 6F1283C072D0014...
0	0	0	0103000020407100000100000006000000D34D6210DB6F014...
-45.6	0	0	0103000020407100000100000009000000046B6F3FD0C2D014...
0	0	0	0103000020407100000100000000A000000C74B3789BE42014...

Stetl Config BRK to PostGIS

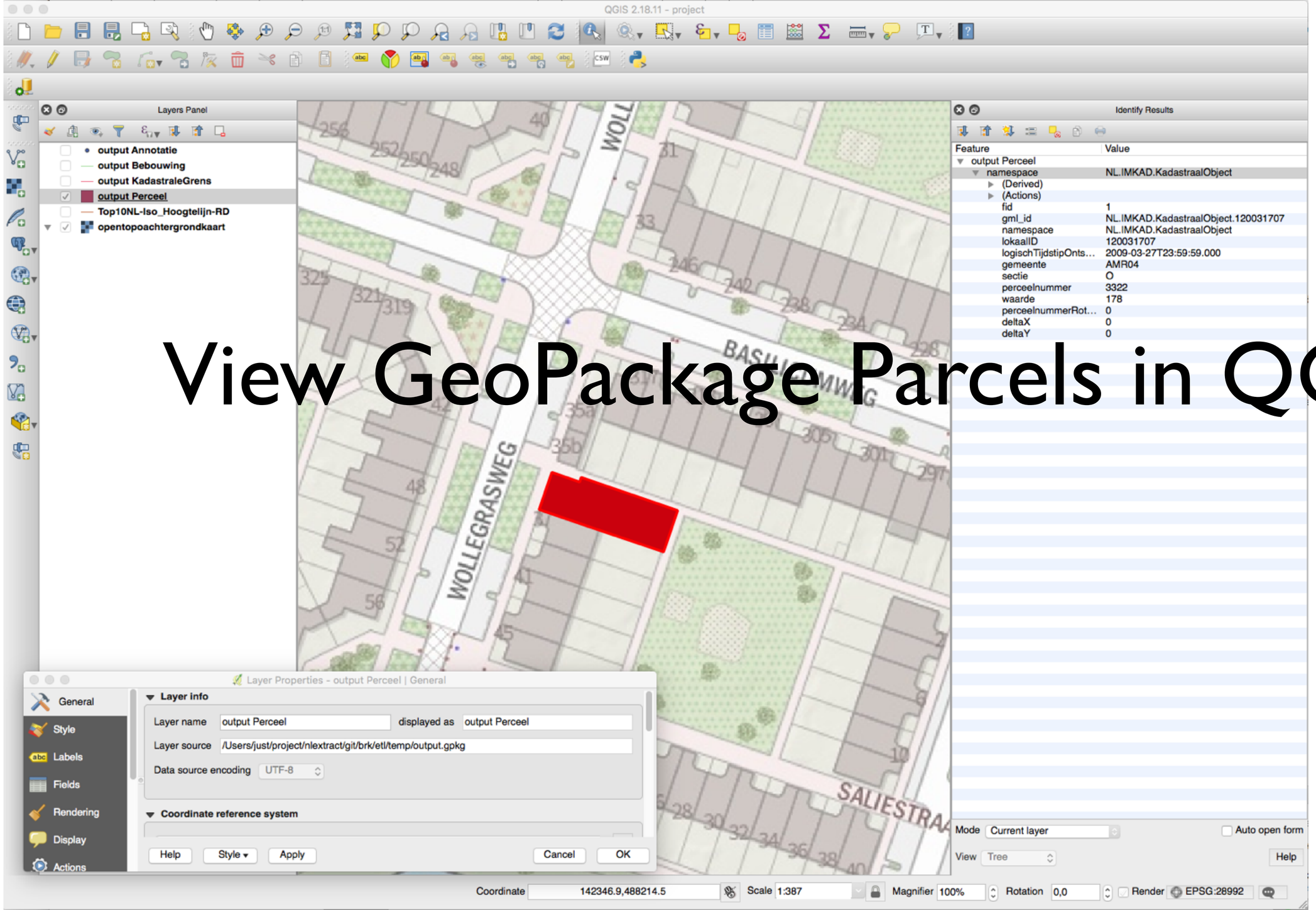


Stetl Config BRK to GeoPackage



Stetl Config - BRK .zip to GeoPackage

```
[etl]
chains = input_zip_file|extract_zip_file|output_ogr2ogr_gpkg

# The source input ZIP-file(s) from dir, producing 'records' with ZIP file name and inner file names
[input_zip_file]
class=inputs.fileinput.ZipFileInput
file_path = {input_dir}
filename_pattern = {zip_files_pattern}
name_filter = {filename_match}

# Filter to extract a ZIP file one by one to a temporary location
[extract_zip_file]
class=filters.zipfileextractor.ZipFileExtractor
file_path = {temp_dir}/fromzip-tmp.gml

[output_ogr2ogr_gpkg]
class = outputs.execoutput.Ogr2OgrExecOutput
# destination format: OGR vector format name
dest_format = GPKG
# destination datasource: path to .gpkg output file
dest_data_source = "{temp_dir}/output.gpkg"
# layer creation options will only be added to ogr2ogr on first run
lco = -lco SPATIAL_INDEX=YES -lco PRECISION=NO
# spatial_extent, translates to -spat xmin ymin xmax ymax
spatial_extent = {spatial_extent}
# gfs template
gfs_template = {gfs_template}
# miscellaneous ogr2ogr options
options = -append -gt 65536 {multi_opts} --config CPL_ZIP_ENCODING CP437
```



View GeoPackage Parcels in QGIS

BRK .zip to both PostGIS + GeoPackage

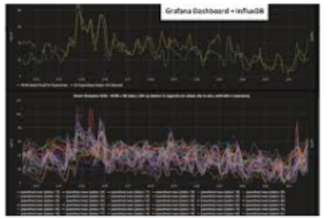
```
[etl]
chains = input_zip_file|extract_zip_file|(output_ogr2ogr_postgis) (output_ogr2ogr_gpkg)

# The source input ZIP-file(s) from dir, producing 'records' with ZIP file name and inner file names
[input_zip_file]
class=inputs.fileinput.ZipFileInput
file_path = {input_dir}
filename_pattern = {zip_files_pattern}
name_filter = {filename_match}

# Filter to extract a ZIP file one by one to a temporary location
[extract_zip_file]
class=filters.zipfileextractor.ZipFileExtractor
file_path = {temp_dir}/fromzip-tmp.gml

[output_ogr2ogr_postgis]
class = outputs.execoutput.Ogr2OgrExecOutput
# destination format: OGR vector format name
dest_format = PostgreSQL
.
.
[output_ogr2ogr_gpkg]
class = outputs.execoutput.Ogr2OgrExecOutput
# destination format: OGR vector format name
dest_format = GPKG
# destination datasource: path to .gpkg output file
dest_data_source = "{temp_dir}/output.gpkg"
.
.
```

Case 2 - Smart Emission

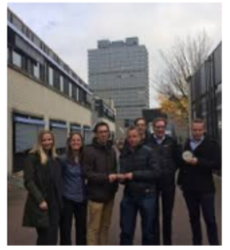






Het ‘Smart Emission’ project draait om het in kaart brengen van luchtkwaliteit, geluid, trillingen en meteorologische indicatoren in de stad op een fijnmazig schaalniveau, door inwoners met zogenoemde burger-sensor-netwerken. De pilotstudie richt zich op de stad Nijmegen (binnen de gemeentegrenzen van de stad). Hier zal een testbed worden ingericht waar het projectteam, in samenwerking met inwoners de lokale variatie in luchtkwaliteit (NO2, CO, CO2, O3), geluid en trillingen in kaart brengen met behulp van betaalbare sensoren.



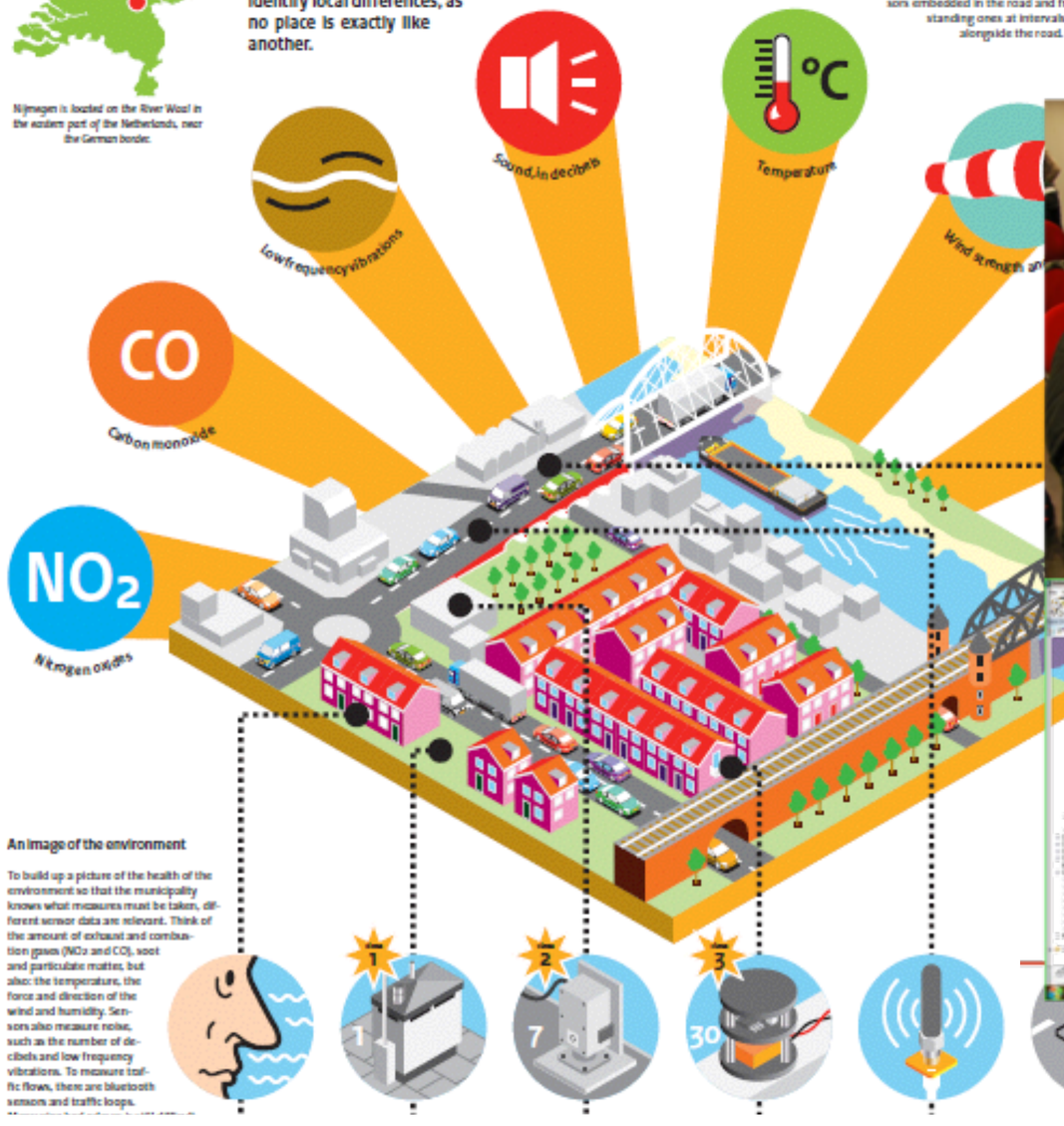


Case: Environmental health in Nijmegen

Nijmegen is located on the River Waal in the eastern part of the Netherlands, near the German border.

The environment is very important for people's health. That's why standards are set for the concentrations of pollutants. Sensors measure whether we keep within these standards using national monitoring networks. In addition, several municipalities and regions have their own sensors to identify local differences, as no place is exactly like another.

Nijmegen also monitors local environmental quality. With the advent of a new bridge and the construction of a ring road, the traffic situation in the western part of Nijmegen has changed. Developments in the port and the industrial area by the River Waal have been made, and residents in the nearby neighbourhood are worried about the health of their environment. The municipality is taking their concerns seriously and has placed sensors in the neighbourhood to measure the air quality and noise level. Nijmegen also wants to collate reports about bad odours. To monitor traffic flows, the municipality uses sensors embedded in the road and free standing ones at intervals alongside the road.



An image of the environment

To build up a picture of the health of the environment so that the municipality knows what measures must be taken, different sensor data are relevant. Think of the amount of exhaust and combustion gases (NO_2 and CO), soot and particulate matter, but also: the temperature, the force and direction of the wind and humidity. Sensors also measure noise, such as the number of decibels and low frequency vibrations. To measure traffic flows, there are bluetooth sensors and traffic loops.

matter sensor units and, as part of a research project, the Radboud university has distributed a 'swarm' of thirty simple sensors among residents. One of the questions posed by this research is

From Pilot to Platform

Radboud Universiteit

GEMEENTE Nijmegen

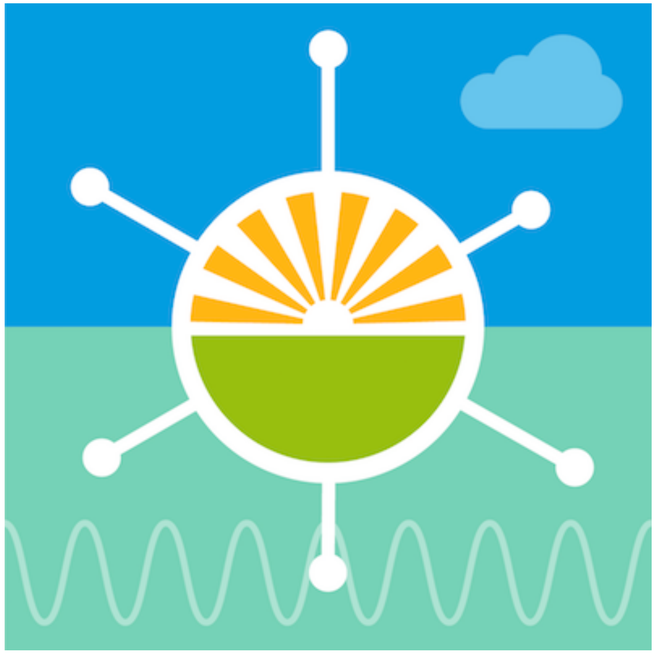
powered by pdok kadaster

GEONOVUM

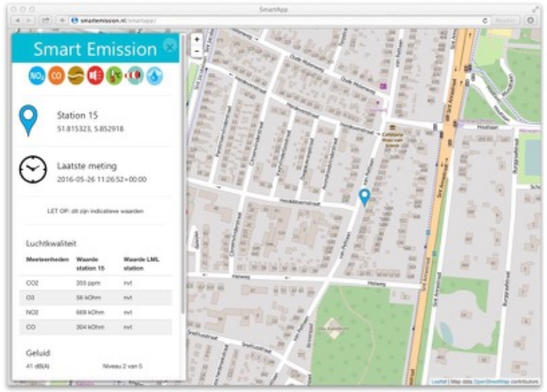
Intemo
INNOVATIVE TECHNOLOGY IN MOTION

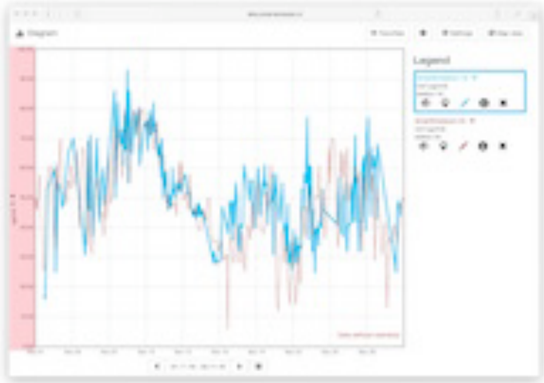
CityGIS
PUBLIC SAFETY

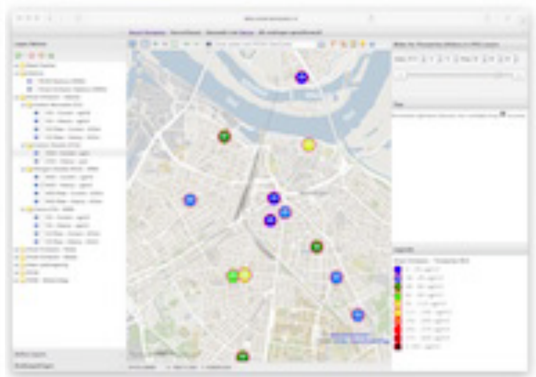
Rijksinstituut voor Volksgezondheid en Milieu
Ministerie van Volksgezondheid, Welzijn en Sport



Viewers





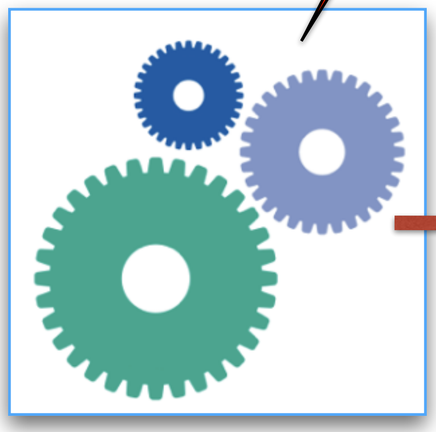


Services

Stetl!

ETL

Sensors

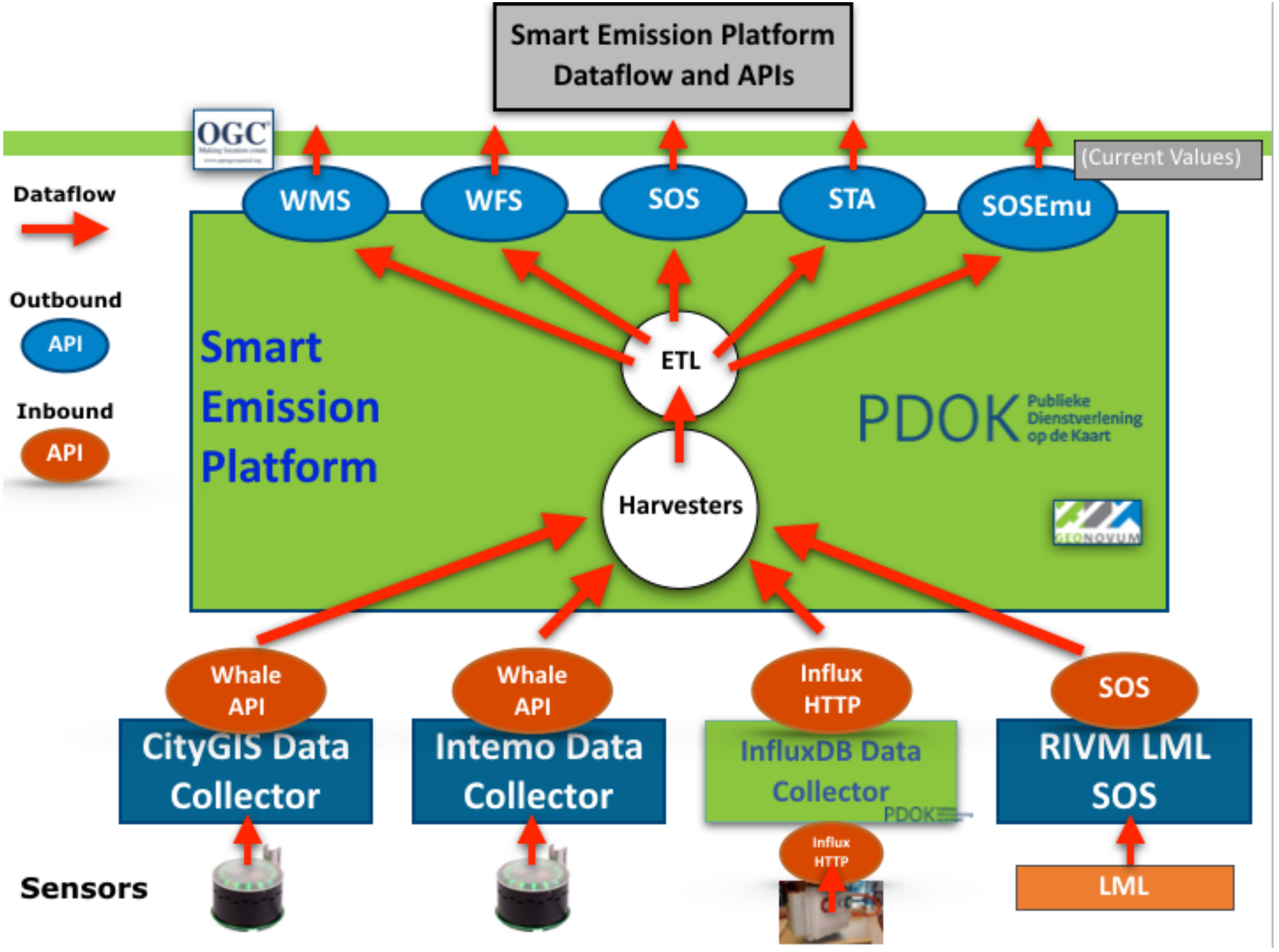


Smart Emission Platform

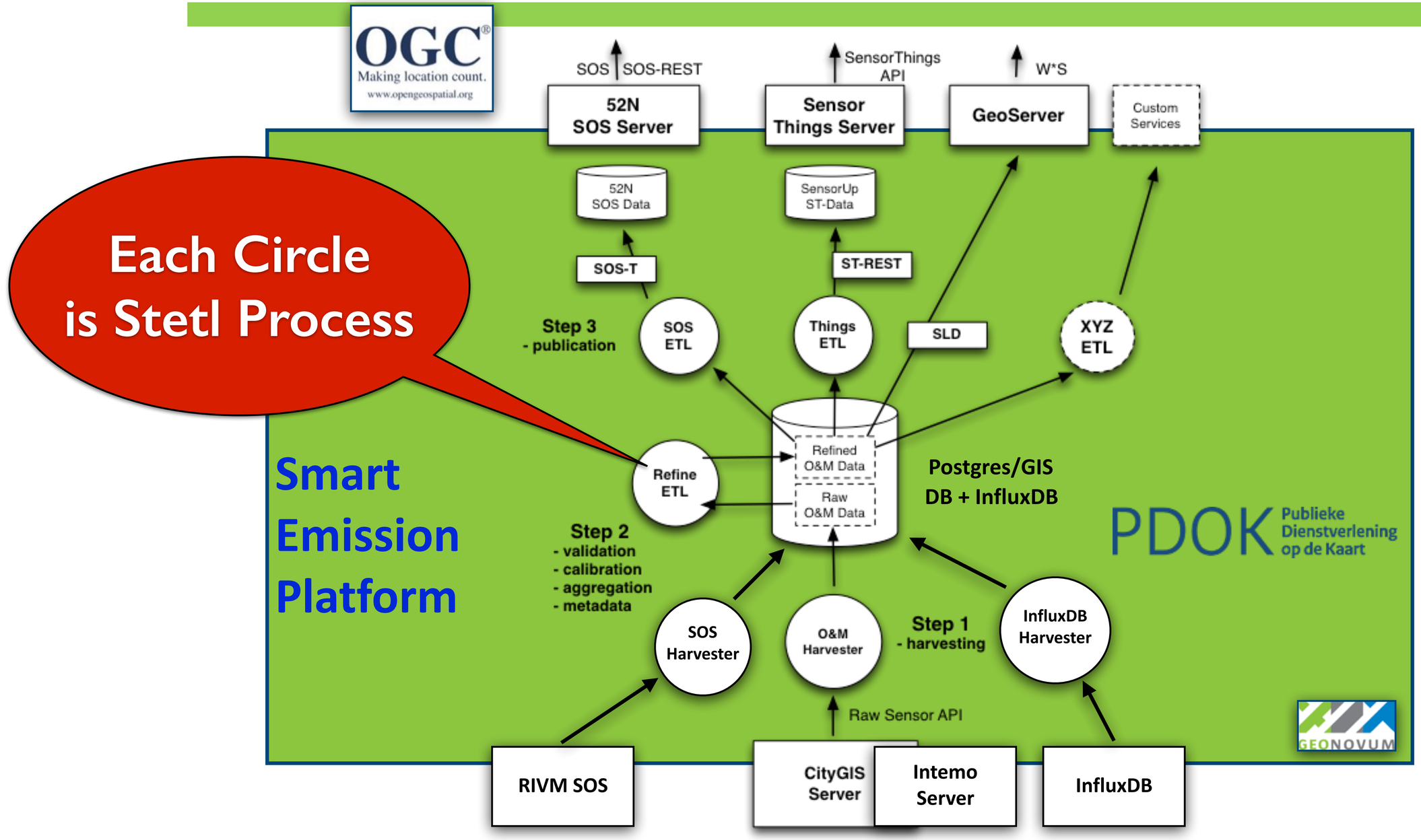
Smart Emission Platform Summary

- Air Quality + Noise + Meteo Sensors hosted by citizens
- Raw sensor data **Harvesting** to SE Platform
- **Validation**, Calibration, Aggregation
- Publication to Web Services (WMS, WFS, SOS, SensorThings API)
- data.smartemission.nl

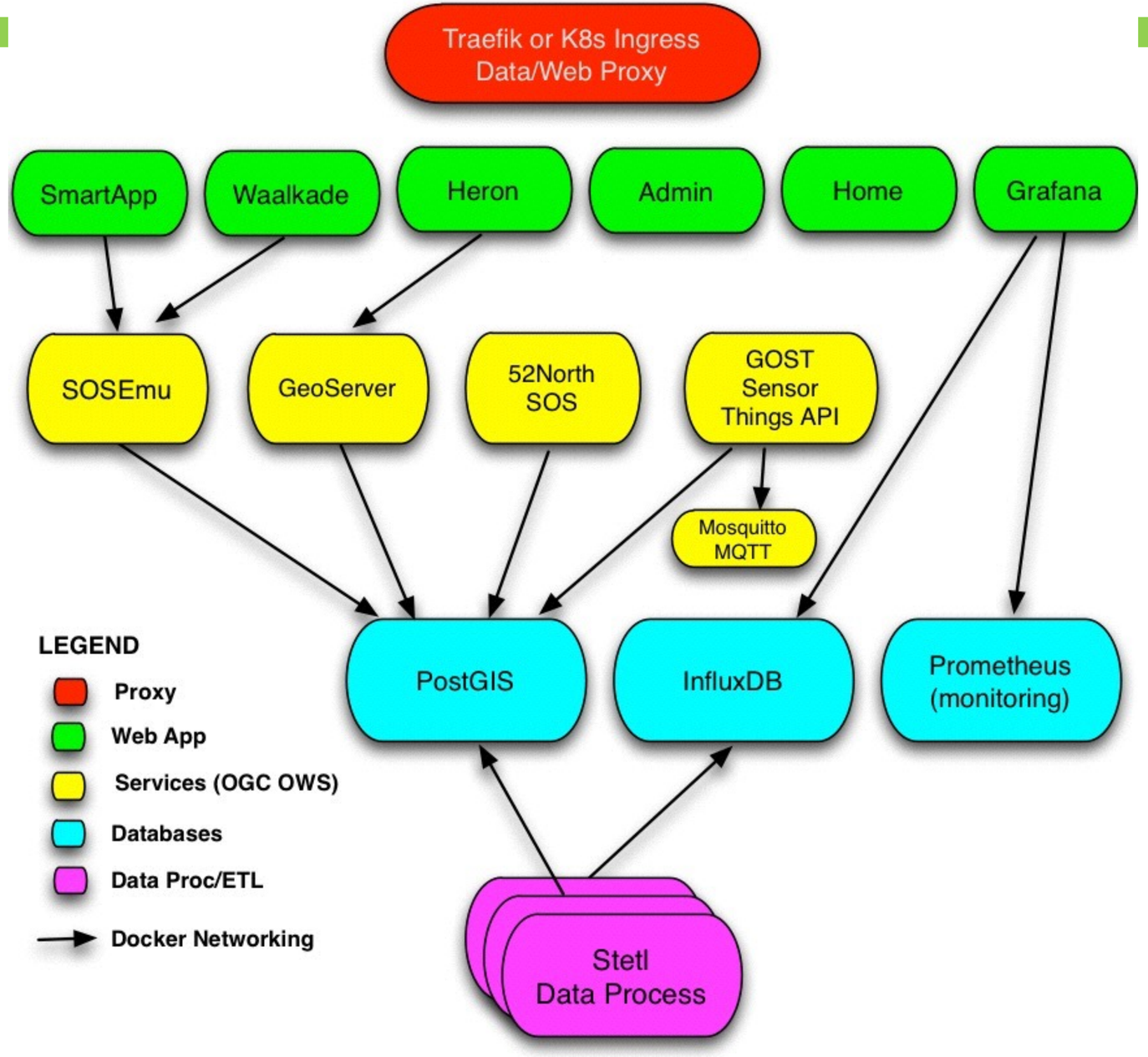
All ETL
(Harvesting, Validation, Calibration, Aggregation, Publication)
done with Stetl

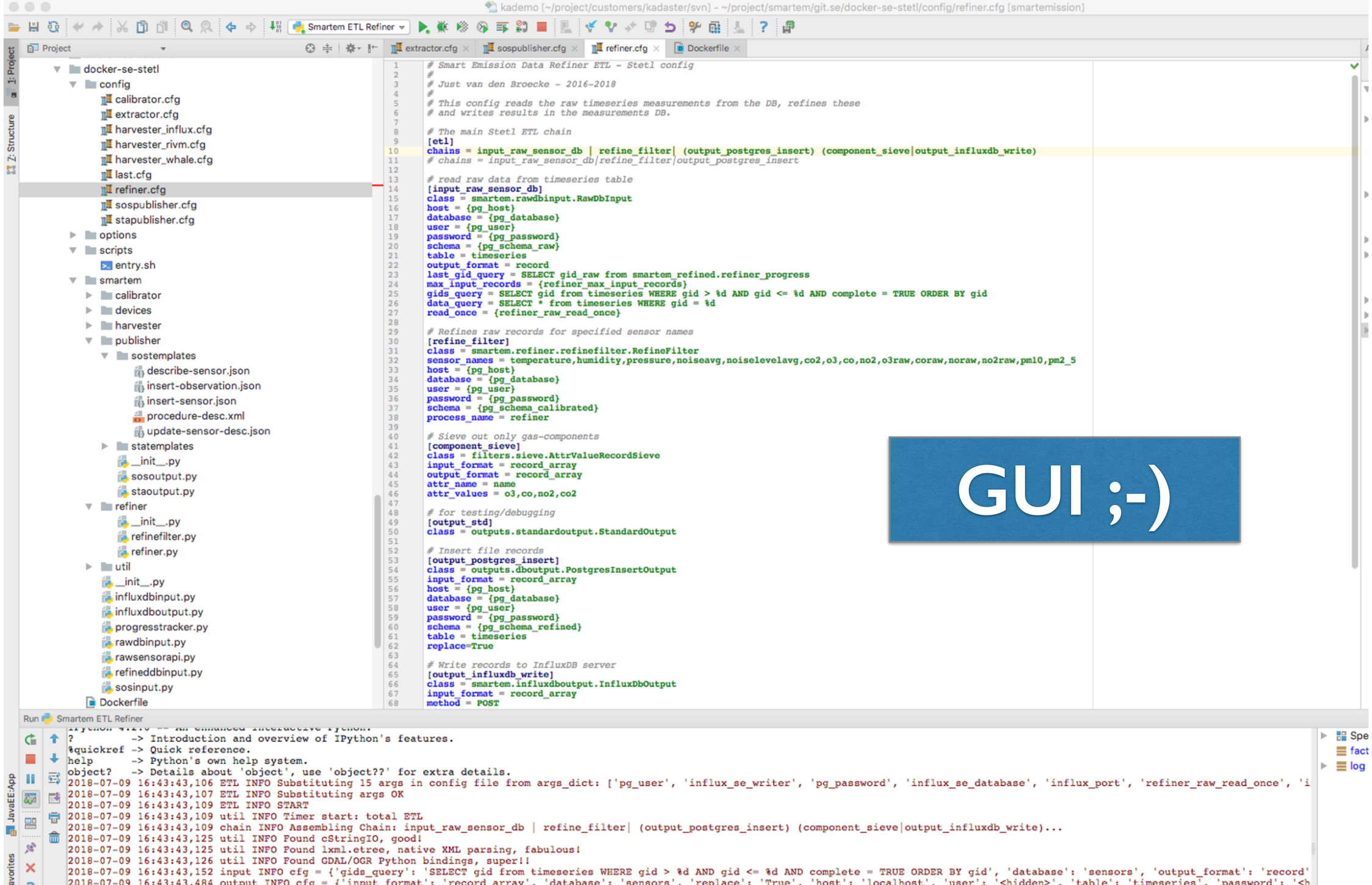

Smart Emission Platform
Dataflow and APIs
OGC
(Current Values)
Dataflow
Outbound
API
Inbound
API
WMS
WFS
SOS
STA
SOSEmu
Smart Emission Platform
ETL
Harvesters
PDOK
Publieke Dienstverlening op de Kaart
GEONOVUM
Whale API
Whale API
Influx HTTP
SOS
CityGIS Data Collector
Intemo Data Collector
InfluxDB Data Collector
PDOK
RIVM LML SOS
Influx HTTP
LML
Sensors

Data Architecture with multi-Step ETL



Smart Emission Docker Deployment




kademo [~/project/customers/kadaster/svn] - ~/project/smartem/git.se/docker-se-stetl/config/refiner.cfg [smartemission]
Smartem ETL Refiner
Project
extractor.cfg
sospublisher.cfg
refiner.cfg
Dockerfile
docker-se-stetl
config
calibrator.cfg
extractor.cfg
harvester_influx.cfg
harvester_rivm.cfg
harvester_whale.cfg
last.cfg
refiner.cfg
sospublisher.cfg
stapublisher.cfg
options
scripts
entry.sh
smartem
calibrator
devices
harvester
publisher
sostemplates
describe-sensor.json
insert-observation.json
insert-sensor.json
procedure-desc.xml
update-sensor-desc.json
statemplates
__init__.py
sosoutput.py
staoutput.py
refiner
__init__.py
refinefilter.py
refiner.py
util
__init__.py
influxdbinput.py
influxdboutput.py
progresstracker.py
rawdbinput.py
rawsensorapi.py
refineddbinput.py
sosinput.py
Dockerfile
Smart Emission Data Refiner ETL - Stetl config
#
Just van den Broecke - 2016-2018
#
This config reads the raw timeseries measurements from the DB, refines these
and writes results in the measurements DB.
The main Stetl ETL chain
[etl]
chains = input_raw_sensor_db | refine_filter| (output_postgres_insert) (component_sieve|output_influxdb_write)
# chains = input_raw_sensor_db|refine_filter|output_postgres_insert
# read raw data from timeseries table
[input_raw_sensor_db]
class = smartem.rawdbinput.RawDbInput
host = {pg_host}
database = {pg_database}
user = {pg_user}
password = {pg_password}
schema = {pg_schema_raw}
table = timeseries
output_format = record
last_gid_query = SELECT gid_raw from smartem_refined.refiner_progress
max_input_records = {refiner_max_input_records}
gids_query = SELECT gid from timeseries WHERE gid > %d AND gid <= %d AND complete = TRUE ORDER BY gid
data_query = SELECT * from timeseries WHERE gid = %d
read_once = {refiner_raw_read_once}
# Refines raw records for specified sensor names
[refine_filter]
class = smartem.refiner.refinefilter.RefineFilter
sensor_names = temperature,humidity,pressure,noiseavg,noiselevelavg,co2,o3,co,no2,o3raw,coraw,noraw,no2raw,pm10,pm2_5
host = {pg_host}
database = {pg_database}
user = {pg_user}
password = {pg_password}
schema = {pg_schema_calibrated}
process_name = refiner
# Sieve out only gas-components
[component_sieve]
class = filters.sieve.AttrValueRecordSieve
input_format = record_array
output_format = record_array
attr_name = name
attr_values = o3,co,no2,co2
# for testing/debugging
[output_std]
class = outputs.standardoutput.StandardOutput
# Insert file records
[output_postgres_insert]
class = outputs.dboutput.PostgresInsertOutput
input_format = record_array
host = {pg_host}
database = {pg_database}
user = {pg_user}
password = {pg_password}
schema = {pg_schema_refined}
table = timeseries
replace=True
# Write records to InfluxDB server
[output_influxdb_write]
class = smartem.influxdboutput.InfluxDbOutput
input_format = record_array
method = POST
GUI ;-)
Run Smartem ETL Refiner
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
2018-07-09 16:43:43,106 ETL INFO Substituting 15 args in config file from args_dict: ['pg_user', 'influx_se_writer', 'pg_password', 'influx_se_database', 'influx_port', 'refiner_raw_read_once', 'i
2018-07-09 16:43:43,107 ETL INFO Substituting args OK
2018-07-09 16:43:43,109 ETL INFO START
2018-07-09 16:43:43,109 util INFO Timer start: total ETL
2018-07-09 16:43:43,109 chain INFO Assembling Chain: input_raw_sensor_db | refine_filter| (output_postgres_insert) (component_sieve|output_influxdb_write)...
2018-07-09 16:43:43,125 util INFO Found cStringIO, good!
2018-07-09 16:43:43,125 util INFO Found lxml.etree, native XML parsing, fabulous!
2018-07-09 16:43:43,126 util INFO Found GDAL/OGR Python bindings, super!!
2018-07-09 16:43:43,152 input INFO cfg = {'gids_query': 'SELECT gid from timeseries WHERE gid > %d AND gid <= %d AND complete = TRUE ORDER BY gid', 'database': 'sensors', 'output_format': 'record'
fact
log

1. Dockerfile

```
FROM geopython/stetl:1.2

# Use the standard Stetl Docker Image with some additional packages
LABEL maintainer="Just van den Broecke <justb4@gmail.com>"

ENV ADD_PYTHON_DEB_PACKAGES="python-dev python-scipy python-seaborn python-matplotlib"  \
    ADD_PYTHON_PIP_PACKAGES="wheel Geohash influxdb scikit-learn==0.18"

# Not now see: https://github.com/pypa/pip/issues/5240
# RUN pip install --upgrade pip
RUN apt-get update && apt-get --no-install-recommends install  -y \
         ${ADD_PYTHON_DEB_PACKAGES} \

    && pip install ${ADD_PYTHON_PIP_PACKAGES} \
    && apt-get purge -y python-dev \
    && apt autoremove -y  \
    && rm -rf /var/lib/apt/lists/*


# Copy relevant files to work dir
ADD config /work/config
ADD smartem /work/smartem
ADD options/example.args /work/options/default.args
ADD scripts /work/scripts

WORKDIR /work

ENTRYPOINT ["/work/scripts/entry.sh"]

```

2. entry.sh

```
#!/bin/bash
#
# Generic ETL entry: override with specific ETL config and args file.
#

WORK_DIR=/work

cd ${WORK_DIR}

# Shorthand
function log() {
    echo "entry.sh: $1"
}

# Shorthand
function error() {
    log "$@"
    exit -1
}

# ETL Process can be specified via env var or argument
if [ "${ETL_PROCESS}x" = "x" ];
then
    ETL_PROCESS="$1"
fi

# ETL Process can be specified via env var or argument
if [ "${ETL_PROCESS}x" = "x" ];
then
    error "Error: no ETL process specified"
fi

ETL_CONFIG_FILE=config/${ETL_PROCESS}.cfg
if [ ! -f ${ETL_CONFIG_FILE} ]
then
    error "Error: cannot find ETL config file: ${ETL_CONFIG_FILE}"
else
    log "OK using config file: ${ETL_CONFIG_FILE}..."
fi

# Start Stetl with config file
export PYTHONPATH=${WORK_DIR}:${PYTHONPATH}
stetl -c ${ETL_CONFIG_FILE} -a options/default.args

```

3. Run ETL Process

```
$ docker run -t smartemission/se-stetl:1.0.13 refiner
```

Thank You !

www.stetl.org

github.com/geopython/stetl