Как организовать ETL с Node.js в serverless-архитектуре

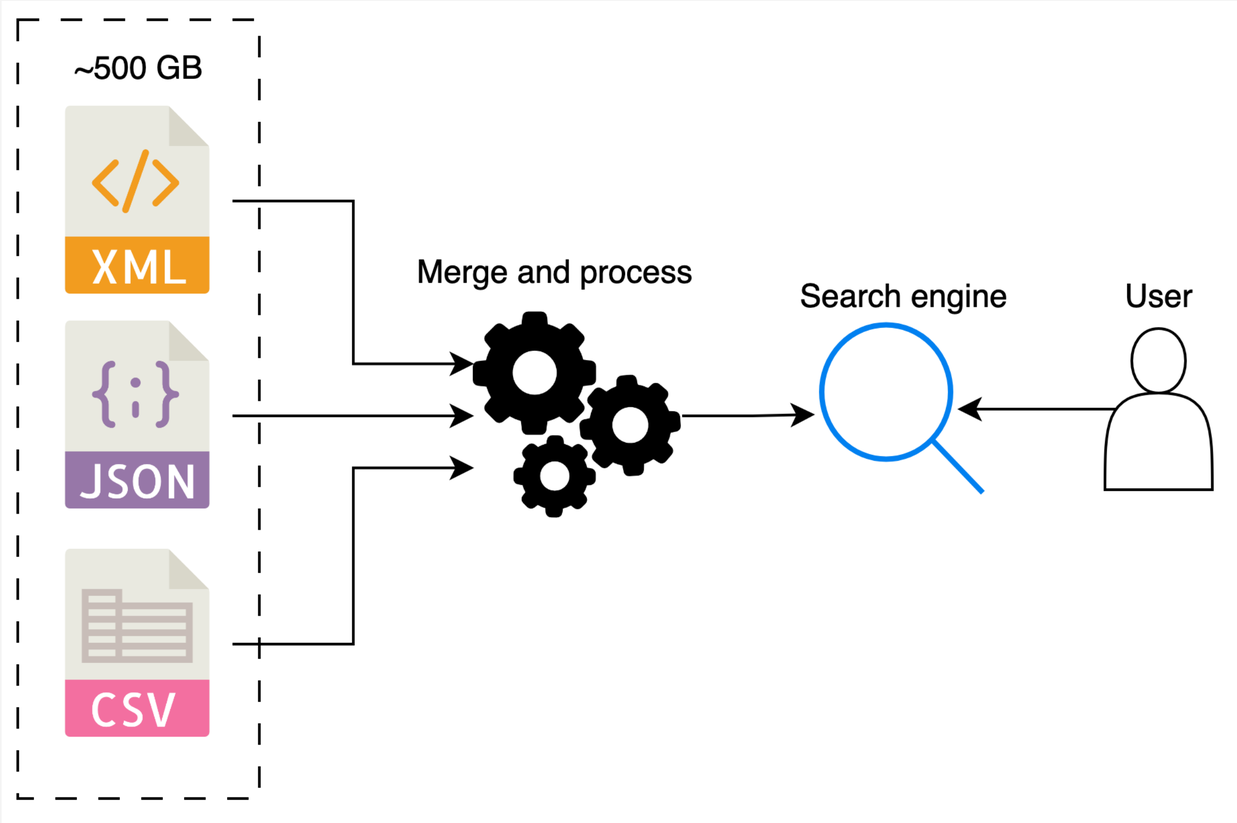
Обработка больших объемов данных с Node.js в инфраструктуре AWS

Dzmitry Krystsek

Основано на реальных событиях

Дисклеймер: Все имена главных героев изменены, данные заменены на не реальные а сама история немного приукрашена что бы вы не испытали всю ту боль что и мы.

Задача



У нас есть 500 гигабайт данных которые обновляются каждый день и мы должны их объединить, преобразовать и добавить в ответ пользователя.

Упрощенная версия

Complete OSM Data

Latest Weekly Planet XML File (**torrent**) (**RSS**)
126 GB, created 2 days ago.
md5: 2e33492905dc2d830c8ce16f4a71bc48.

Так как мы не можем показать реальных данных давайте изменим наш пример. И поговорим про наш пример с osm xml

OSM

```
<relation id="15793613" version="1">
  <member type="way" ref="754571017" role="outer"/>
  <member type="way" ref="1167683726" role="inner"/>
  <member type="way" ref="1167683724" role="inner"/>
  <tag k="landuse" v="forest"/>
  <tag k="type" v="multipolygon"/>
</relation>
<way id="23388940" version="15">
  <nd ref="253241462"/>
  <nd ref="253241463"/>
  <nd ref="253241464"/>
  <nd ref="253241465"/>
  <nd ref="253241462"/>
  <tag k="addr:housenumber" v="17"/>
  <tag k="addr:postcode" v="220021"/>
  <tag k="addr:street" v="Цэнтральная вуліца"/>
  <tag k="building" v="house"/>
  <tag k="building:levels" v="2"/>
</way>
<node id="253241462" lat="53.8701973" lon="27.6511786"/>
<node id="253241463" lat="53.8701632" lon="27.6513107"/>
<node id="253241464" lat="53.8702414" lon="27.6513687"/>
<node id="253241465" lat="53.8702754" lon="27.6512366"/>
```

Transform

```
{
  "id": 23388940,
  "tags": {
    "addr:housenumber": "17",
    "building:levels": 2,
    "addr:street": "Цэнтральная вуліца",
    "building": "house"
  },
  "geometry": [
    [27.6511786, 53.8701973],
    [27.6513107, 53.8701632],
    [27.6513687, 53.8702414],
    [27.6512366, 53.8702754],
    [27.6511786, 53.8701973]
  ],
  "center": [27.403958, 53.843655],
  "square": 88.5
}
```

Как это сделать ? (ETL)

Complete OSM Data

[Latest Weekly Planet XML File](#) ([torrent](#)) ([RSS](#))
126 GB, created 2 days ago.
md5: 2e33492905dc2d830c8ce16f4a71bc48.

Extract

```
<relation id="15793613" version="1">
  <member type="way" ref="754571017" role="outer"/>
  <member type="way" ref="1167683726" role="inner"/>
  <member type="way" ref="1167683724" role="inner"/>
  <tag k="landuse" v="forest"/>
  <tag k="type" v="multipolygon"/>
</relation>
<way id="23388940" version="15">
  <nd ref="253241462"/>
  <nd ref="253241463"/>
  <nd ref="253241464"/>
  <nd ref="253241465"/>
  <nd ref="253241462"/>
  <tag k="addr:housenumber" v="17"/>
  <tag k="addr:postcode" v="220021"/>
  <tag k="addr:street" v="Цэнтральная вуліца"/>
  <tag k="building" v="house"/>
  <tag k="building:levels" v="2"/>
</way>
<node id="253241462" lat="53.8701973" lon="27.6511786"/>
<node id="253241463" lat="53.8701632" lon="27.6513107"/>
<node id="253241464" lat="53.8702414" lon="27.6513687"/>
<node id="253241465" lat="53.8702754" lon="27.6512366"/>
```

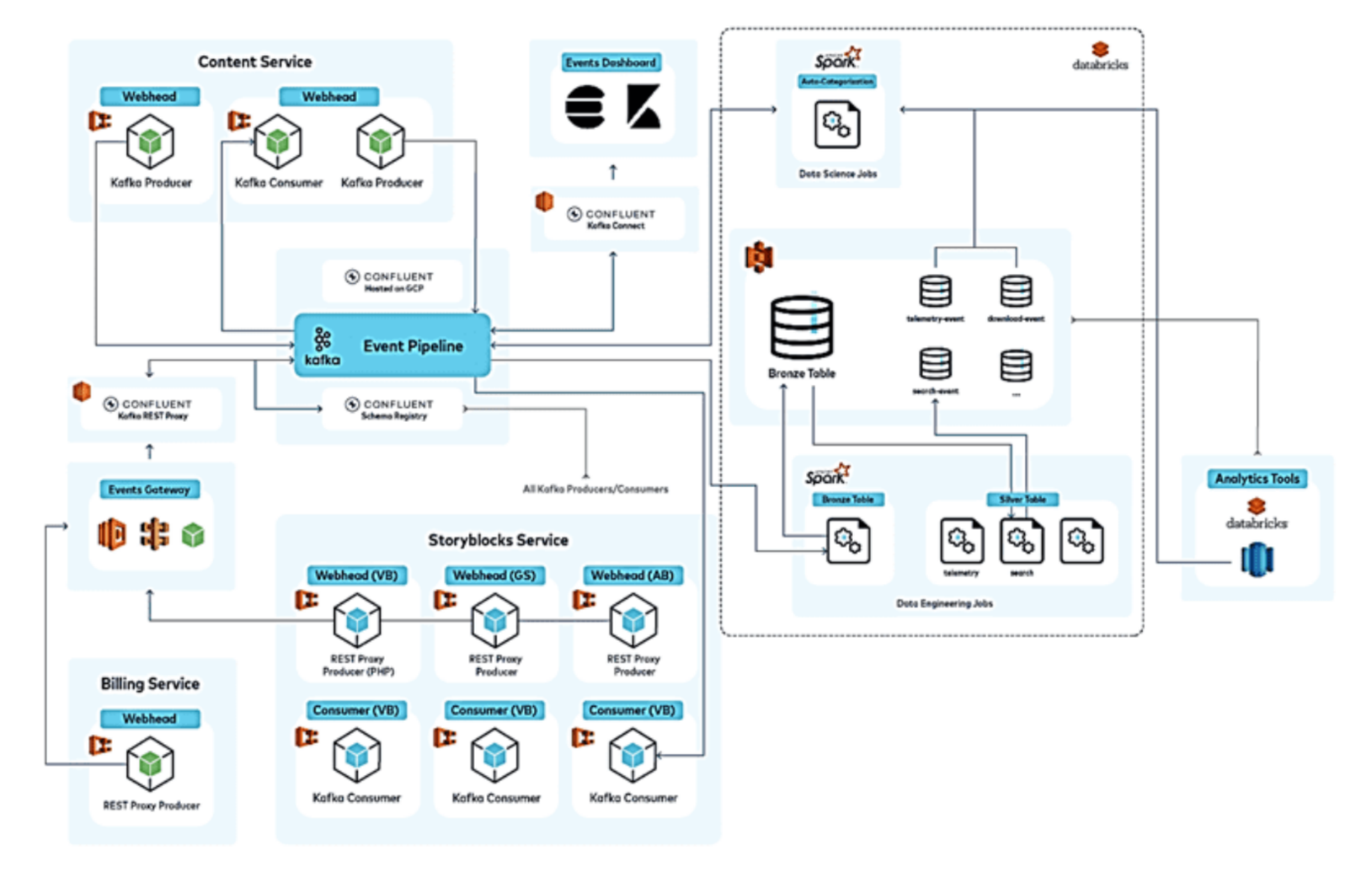
Remove invalid buildings

Transform

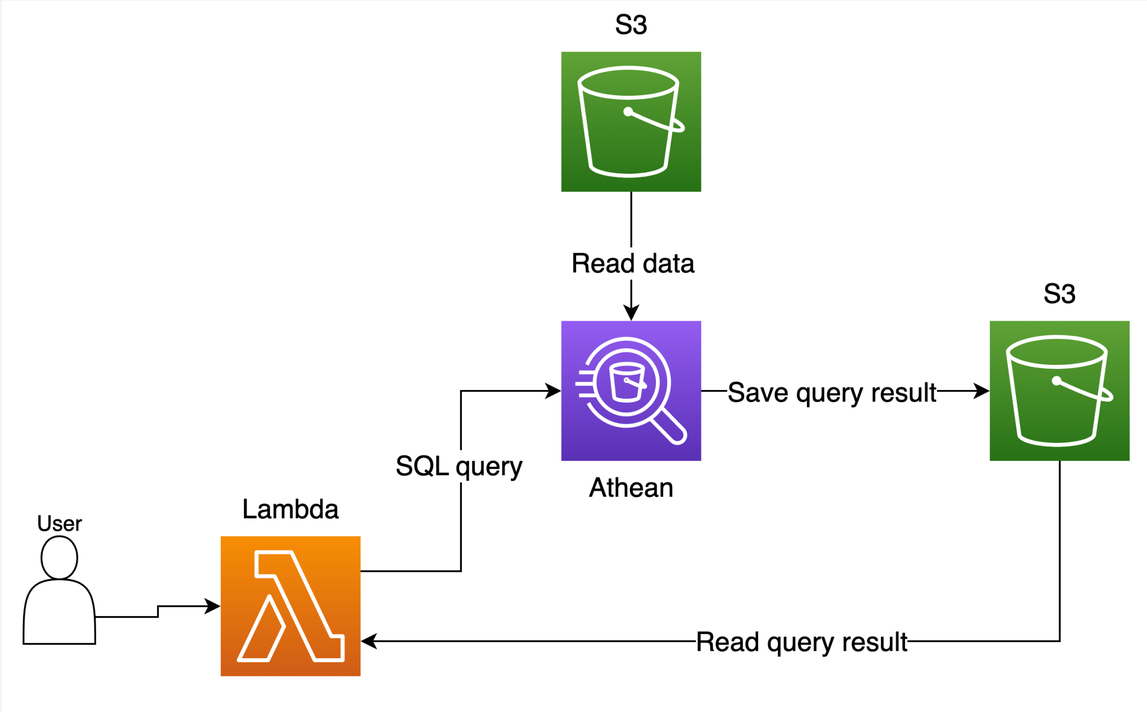
```
{
  "id": 23388940,
  "tags": {
    "addr:housenumber": "17",
    "building:levels": 2,
    "addr:street": "Цэнтральная вуліца",
    "building": "house"
  },
  "geometry": [
    [27.6511786, 53.8701973],
    [27.6513107, 53.8701632],
    [27.6513687, 53.8702414],
    [27.6512366, 53.8702754],
    [27.6511786, 53.8701973]
  ],
  "center": [27.403958, 53.843655],
  "square": 88.5
}
```

Load

Amazon RDS

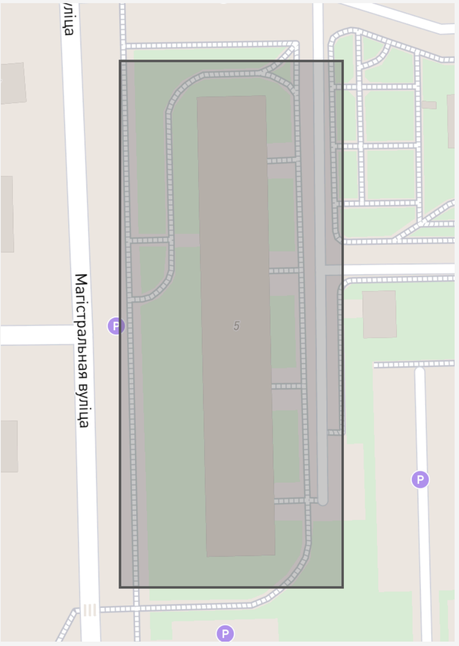

Content Service
Webhead
Kafka Producer
Webhead
Kafka Consumer
Kafka Producer
Events Dashboard
CONFLUENT
Kafka Connect
CONFLUENT
Hosted on GCP
kafka
Event Pipeline
CONFLUENT
Kafka REST Proxy
CONFLUENT
Schema Registry
All Kafka Producers/Consumers
Events Gateway
Storyblocks Service
Webhead (VB)
Webhead (GS)
Webhead (AB)
REST Proxy Producer (PHP)
REST Proxy Producer
REST Proxy Producer
Consumer (VB)
Consumer (VB)
Consumer (VB)
Kafka Consumer
Kafka Consumer
Kafka Consumer
Billing Service
Webhead
REST Proxy Producer
Spark
Auto-Categorization
Data Science Jobs
databricks
Bronze Table
telemetry-event
download-event
search-event
...
Spark
Bronze Table
Silver Table
telemetry
search
Data Engineering Jobs
Analytics Tools
databricks

Demo project



Apache Spark - Сложно,
инфраструктура, мы же js разработчики
Athena
aws glue

Athena query



Query nodes by bbox

```
SELECT *
FROM planet
WHERE 1=1
  AND type = 'node'
  AND lon BETWEEN 27.4036103 AND 27.4042967
  AND lat BETWEEN 53.8431036 AND 53.8441982
```

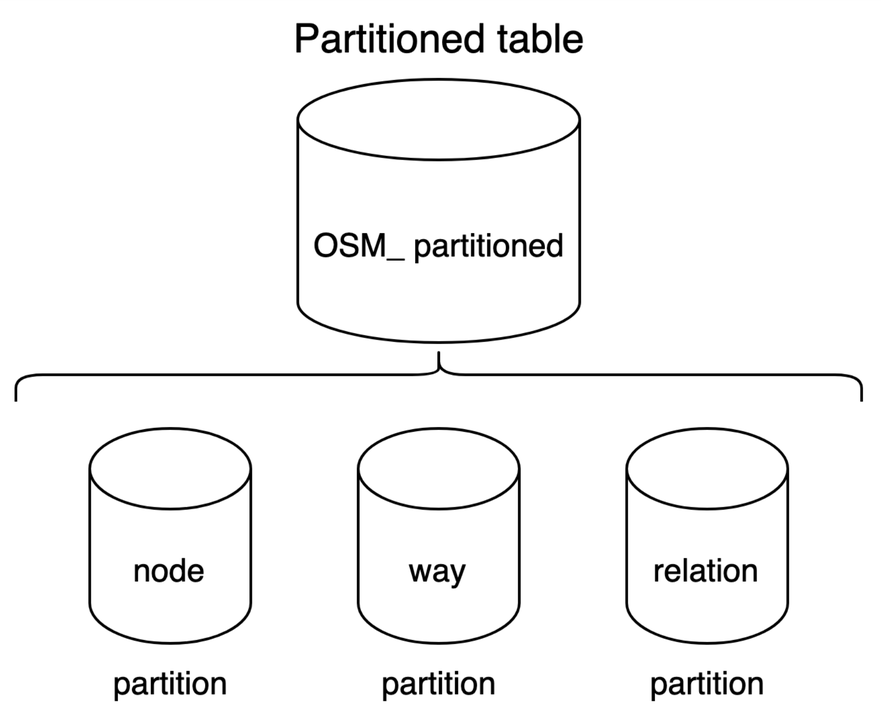
Run time: 11.098 sec, Data scanned: 51.51GB Price: ~0.25$

Query building by bbox

```
WITH nodes AS (
  SELECT id, lon, lat
  FROM planet
  WHERE 1=1
    AND type = 'node'
    AND lon BETWEEN 27.4036103 AND 27.4042967
    AND lat BETWEEN 53.8431036 AND 53.8441982
), buildings AS (
  SELECT id, nds, tags
  FROM planet
  WHERE 1=1
    AND type = 'way'
    AND element_at(tags, 'building') IS NOT NULL
)
SELECT buildings.id, buildings.tags, nodes.lon, nodes.lat
FROM buildings
CROSS JOIN UNNEST(buildings.nds) WITH ORDINALITY t(nd, nd_index)
INNER JOIN nodes on nd.ref = nodes.index
```

Run time: 2 min 15 sec, Data scanned: 82.68GB Price: ~0.41$

Partitions



Athena CTAS

```
CREATE TABLE partition_planet WITH (
  format = 'ORC',
  write_compression = 'ZLIB',
  external_location = 's3://osm/partition_planet/',
  partitioned_by = ARRAY['type', 'lon_p', 'lat_p']
) AS SELECT
  id,
  tags,
  lat,
  lon,
  nds,
  type,
  cast(planet.lon / 10 as integer) as lon_p,
  if(planet.lat > 0, 1, -1) as lat_p
FROM planet
```

☐	type=node/	Folder
☐	type=relation/	Folder
☐	type=way/	Folder

Partitioned athena query

Query nodes by bbox

```
SELECT *
FROM partition_planet
WHERE 1=1
  AND type = 'node'
  AND lon_p = 3
  AND lat_p = 1
  AND lon BETWEEN 27.4036103 AND 27.4042967
  AND lat BETWEEN 53.8431036 AND 53.8441982
```

Run time: 5.65 sec, Data scanned: 2.48GB Price: ~0.0124$

Before Run time: 11.098 sec, Data scanned: 51.51 GB Price: ~0.25$

Improved: Run time - x2, Price - x2

Query building by bbox

```
WITH nodes AS (
  SELECT *
  FROM partition_planet
  WHERE 1=1
  AND type = 'node'
  AND lon_p = 3
  AND lat_p = 1
  AND lon BETWEEN 27.4036103 AND 27.4042967
  AND lat BETWEEN 53.8431036 AND 53.8441982
), buildings AS (
  SELECT *
  FROM partition_planet
  WHERE 1=1
    AND type = 'way'
    AND element_at(tags, 'building') IS NOT NULL
)
SELECT *
FROM buildings
CROSS JOIN UNNEST(buildings.nds) WITH ORDINALITY t(nd, nd_index)
INNER JOIN nodes on nd.ref = nodes.id
```

Run time: 1min 41sec, Data scanned: 29GB Price: ~0.145$

One more attempt

Athena CTAS

```sql
CREATE TABLE planet_buildings
WITH (
  format = 'ORC',
  write_compression = 'ZLIB',
  external_location = 's3://osm/planet_buildings/',
  partitioned_by = ARRAY['lon_p', 'lat_p']
) AS SELECT
  partition_planet.id,
  partition_planet.tags,
  nd.ref AS ref,
  nd_index AS ref_index,
  nodes.lon,
  nodes.lat,
  nodes.lon_p,
  nodes.lat_p
FROM partition_planet as partition_planet
CROSS JOIN UNNEST(partition_planet.nds) WITH ORDINALITY t(nd, nd_index)
INNER JOIN partition_planet as nodes on nd.ref = nodes.id and nodes.type = 'node'
WHERE 1=1
  AND partition_planet.type = 'way'
  AND element_at(partition_planet.tags, 'building') IS NOT NULL
```

Результат

Athena query

```sql
SELECT
  id AS id,
  arbitrary(tags) AS tags,
  ARRAY_AGG(ARRAY[lon, lat] ORDER BY ref_index ASC) AS geometry,
  arbitrary(lon_p) AS lon_p,
  arbitrary(lat_p) AS lat_p
FROM planet_buildings
WHERE 1=1
AND lon_p = 3
AND lat_p = 1
AND lon BETWEEN 27.4036103 AND 27.4042967
AND lat BETWEEN 53.8431036 AND 53.8441982
GROUP BY planet_buildings.id
```

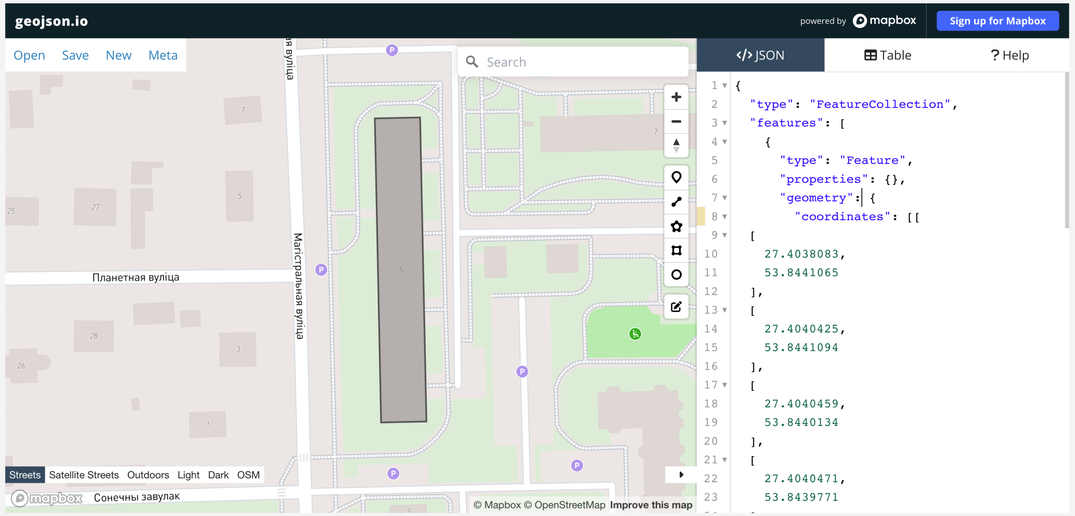
Run time: 2.83 sec, Data scanned: 1.47GB Price: ~0.00735$

Run time: 2 min 15 sec, Data scanned: 82.68GB Price: ~0.41$

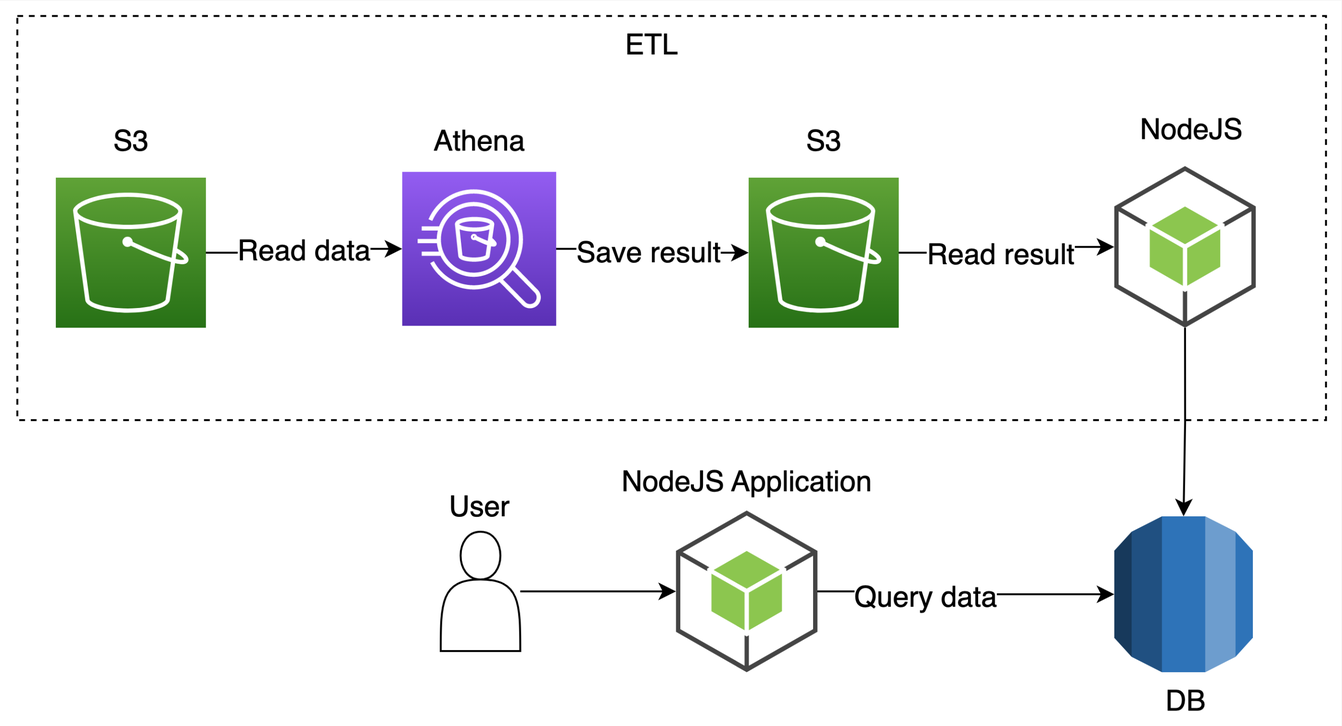
Improved: Run time - x45, Price - x60

Athena result

#	id	tags	geometry	lon_p	lat_p
1	154863108	{addr:housenumber=5, building:levels=9, addr:street=Палеская вуліца, building=apartments}	[[27.4038083, 53.8441065], [27.4040425, 53.8441094], [27.4040459, 53.8440134], [27.4040471, 53.8439771], [27.4040549, 53.8437555], [27.4040631, 53.8435229], [27.4040712, 53.8432918], [27.4040726, 53.8432523], [27.4040750, 53.8431828], [27.4038409, 53.8431799], [27.4038190, 53.8438046], [27.4038167, 53.8438305], [27.4038083, 53.8441065]]	3	1



Production version



Что мы имеем на данный момент?

Partitions

Name	Type
lon_p=-1/	Folder
lon_p=-10/	Folder
lon_p=-11/	Folder
lon_p=-12/	Folder
lon_p=-13/	Folder
lon_p=-14/	Folder

Name	Type
lat_p=-1/	Folder
lat_p=1/	Folder

Partition fiels

20230209_235542_00064_mezsp_02b8162d-2cd9-4305-9c06-a5f564925e1a.gz
20230209_235542_00064_mezsp_173af16f-adda-4793-bccf-b8fa7838eaeb.gz
20230209_235542_00064_mezsp_20e35ae2-3bf8-4d3d-beea-a96a924e15c8.gz
20230209_235542_00064_mezsp_2229fd00-989d-49d5-be17-371dabb55482.gz
20230209_235542_00064_mezsp_343d5073-fa0d-4091-b288-37de33652e66.gz
20230209_235542_00064_mezsp_4f9f1539-e53c-48a4-9d14-5d063061f0b3.gz
20230209_235542_00064_mezsp_5bdbbb52-31f5-485c-8618-c842c5e6fb66.gz
20230209_235542_00064_mezsp_610b3887-1dbc-4451-8cac-b812a3cad320.gz
20230209_235542_00064_mezsp_6442519a-61b8-4755-8167-785f2e5a6169.gz

Как обработать отдельный файл?

building.json.gz
gzip compressed archive - 44 MB

```
1  import { readFile } from 'node:fs/promises';
2  import { gunzipSync } from 'node:zlib';
3  import { formatData } from './format.js';
4  import { filterInvalidData } from './filter.js';
5  import { saveToDatabase } from './database.js';
6
7  const file = await readFile("data.json.gz");
8  const jsonData = gunzipSync(file);
9  const data = JSON.parse(jsonData);
10 const formattedData = formatData(data);
11 const result = filterInvalidData(formattedData);
12 await saveToDatabase(result);
```

Run time: 3.236s
Memory usage: 1194.62 MB

Stream

```
1   import { pipeline } from "node:stream/promises";
2   import { createReadStream } from 'node:fs';
3   import { createGunzip } from 'node:zlib';
4   import ndjson from 'ndjson';
5   import through2Concurrent from "through2-concurrent";
6   import { formatData } from './format.js';
7   import { filterInvalidData } from './filter.js';
8   import { saveToDatabase } from './database.js';
9
10  await pipeline([
11    createReadStream("data.ndjson.gz"),
12    createGunzip(),
13    ndjson.parse(),
14    formatData(),
15    filterInvalidData(),
16    groupStreamObjectsIntoChunks(),
17    through2Concurrent.obj(
18      { maxConcurrency: databasePoolCount },
19      (chunk, enc, callback) => {
20        saveToDatabase(chunk).then(callback);
21      }
22    )
23  ]);
```

```
1   { "center": [27.403958, 53.843655],"square": 88.5 }
2   { "center": [28.553455, 53.424234],"square": 12.5 }
3   { "center": [26.424244, 54.423424],"square": 100.5 }
```

Run time: 2.667s
Memory usage: 23.58 MB

Run time: 3.236s,
Memory usage: 1194.62

Improved: Run time - x1.2,
Memory usage - x50

Worker thread

```javascript
import { pipeline } from "node:stream/promises";
import { createReadStream } from 'node:fs';
import zlib from 'node:zlib';
import through2Concurrent from "through2-concurrent";
import { WorkersPool } from "./workers-pool.js";

const workersPool = new WorkersPool(4);

await pipeline([
  createReadStream("./data/building.ld.geojson.gz"),
  zlib.createGunzip(),
  split(100),
  through2Concurrent.obj(
    { maxConcurrency: workersPool.count },
    (chunk, enc, callback) => {
      workersPool
        .process(chunk)
        .then((res) => {
          buildingsAnalytic.good += res.good;
          buildingsAnalytic.bad += res.bad;
          callback();
        });
    }
  ),
]);
```

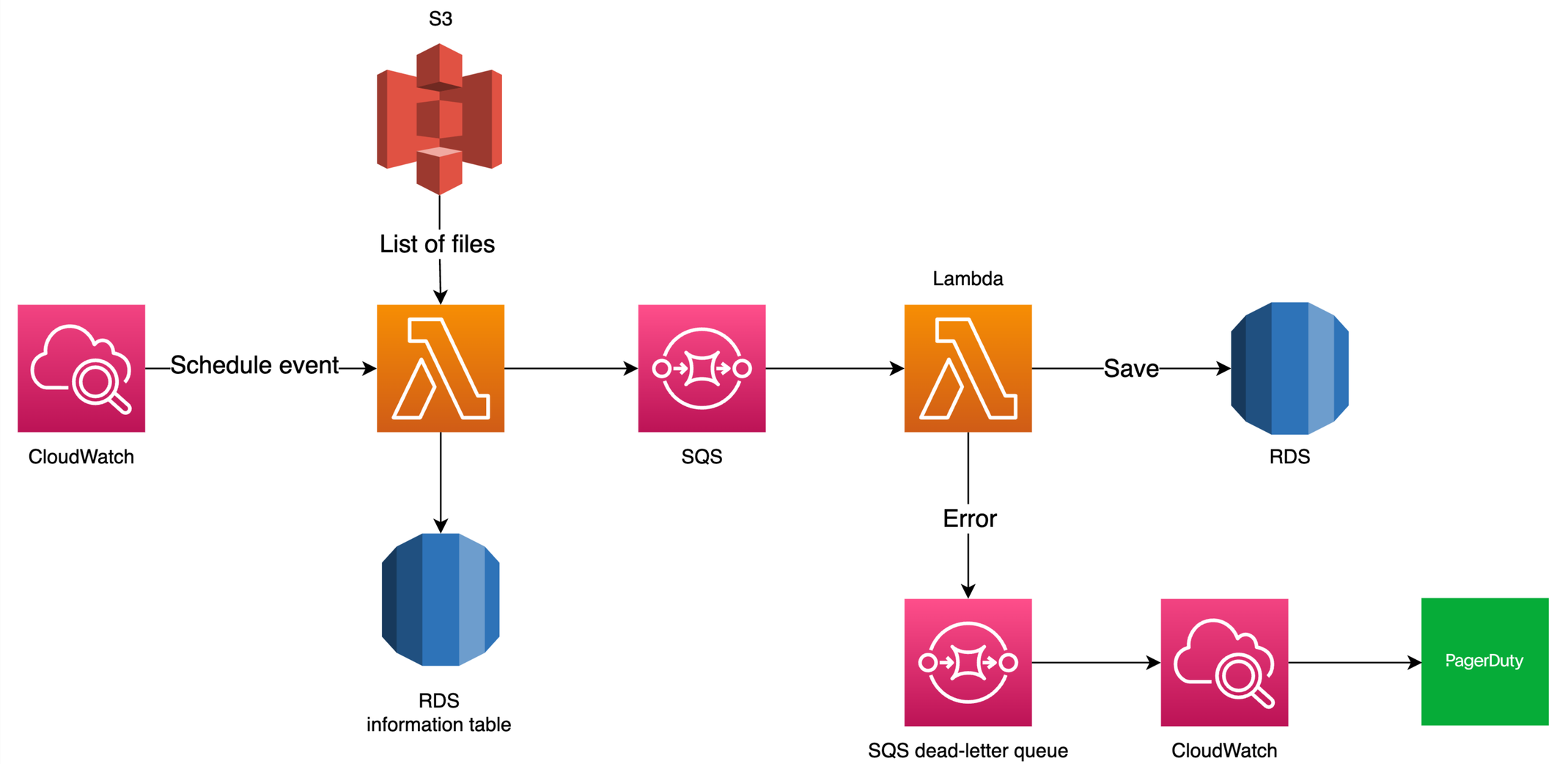
```javascript
export function split({ batchSize = 100 }) {
  let batch = [];
  return new Transform({
    highWaterMark: batchSize,
    objectMode: true,
    transform(chunk, enc, callback) {
      if (batch.length < batchSize) {
        batch.push(chunk);
        callback();
        return;
      }
      // search for delimiter in first chunk
      for (let i = 0; i < chunk.byteLength; i++) {
        if (chunk[i] === delimiter[0]) {
          // ...............
          return callback(null);
        }
      }
      batch.push(chunk);
      callback();
    },
    flush(callback) {
      // ...............
    },
  });
}
```

Run time: 0.732s

Run time: 2.667s,
Memory usage: 1194.62

Improved: Run time - x3.5

Queue


S3
List of files
Lambda
Schedule event
Save
CloudWatch
SQS
RDS
Error
RDS
information table
SQS dead-letter queue
CloudWatch
PagerDuty

Важные моменты по Node.JS

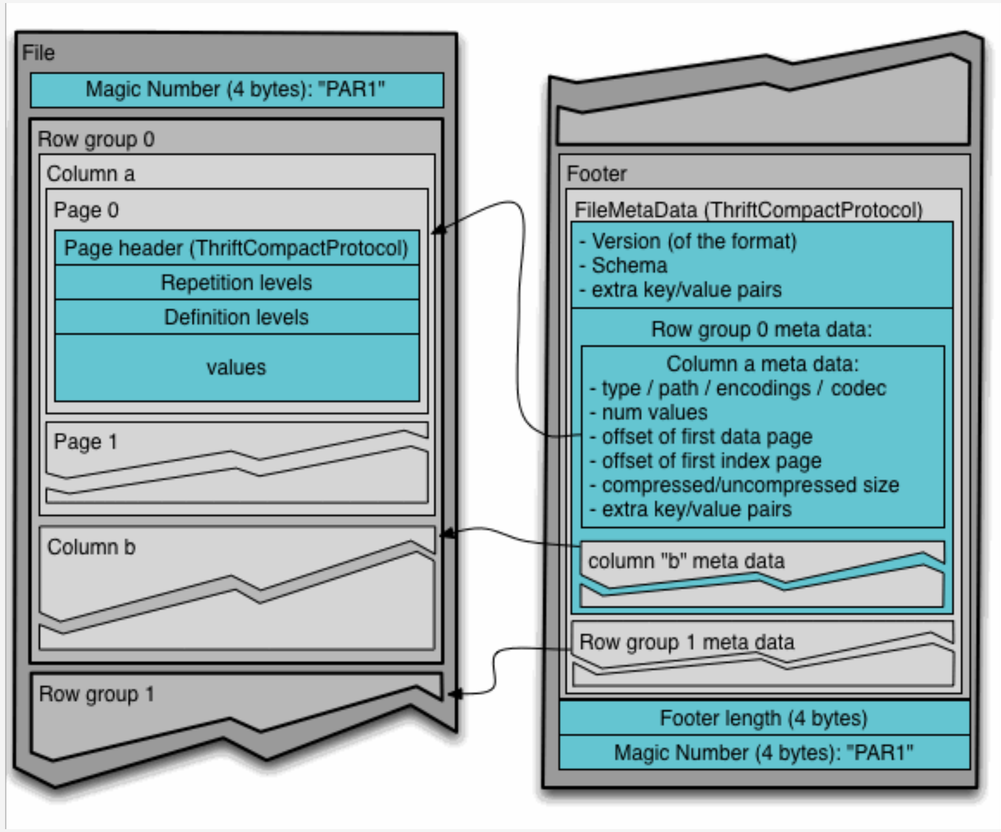
```
1   process.on('unhandledRejection', (up) => {
2     console.log('UnhandledRejection', up);
3     throw up;
4   });
```

Переосмысление

Проблема номер 1: Слишком большие файлы

1. Добавить шаг разбиения файлов.

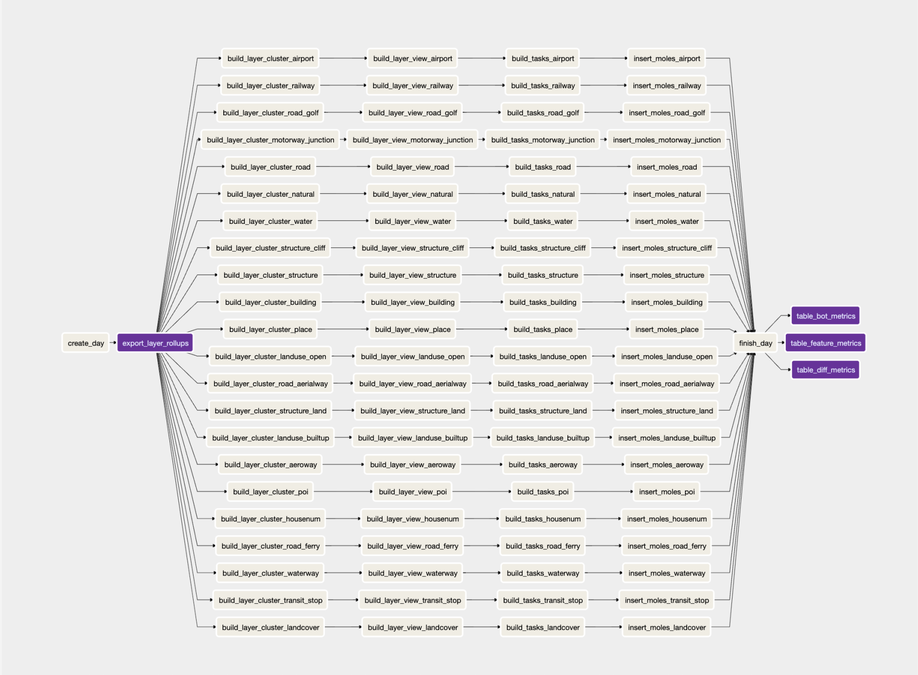
2. Выбрать другой формат данных.



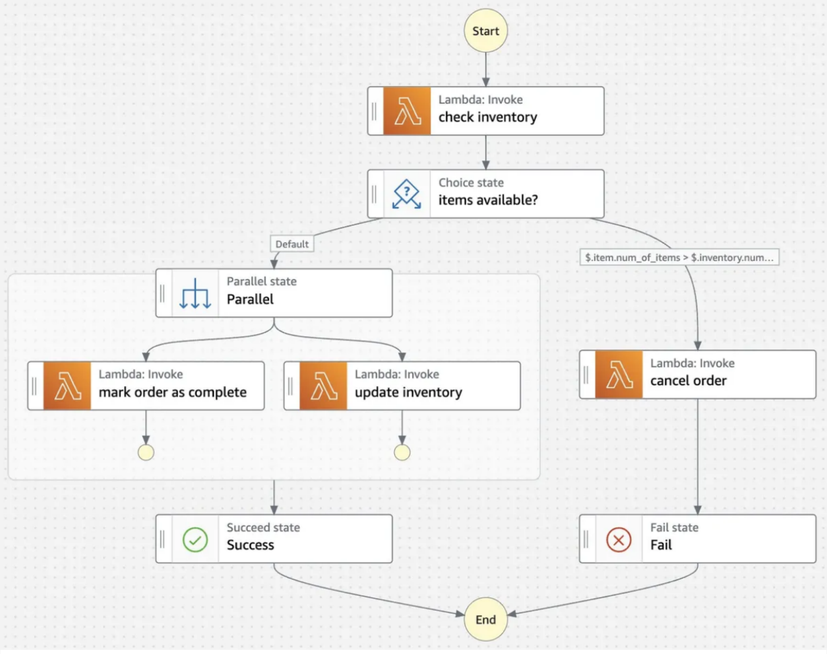
Переосмысление

Проблема номер 1: Как запускать обработку и как отслеживать что все было выполнено?

Apache Airflow



AWS Step Functions



Как это поддерживать и мониторить ?

- Служба уведомлений
- Повторное выполнение кода
- CloudWatch Dashboards
- Мониторинг основных параметров