BOOSTCOM

EVENT SOURCING WITH ELASTICSEARCH

“ES with ES”

by Paweł Świątkowski

Agenda

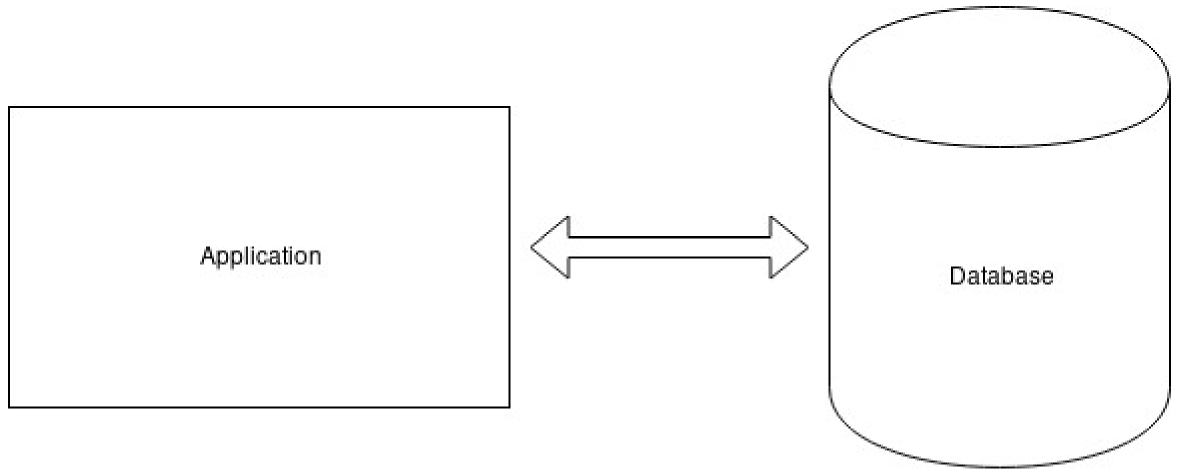
- Event Sourcing – what and why?
- ElasticSearch – what and why?
- Putting it together - how we do it @ Boostcom

About me

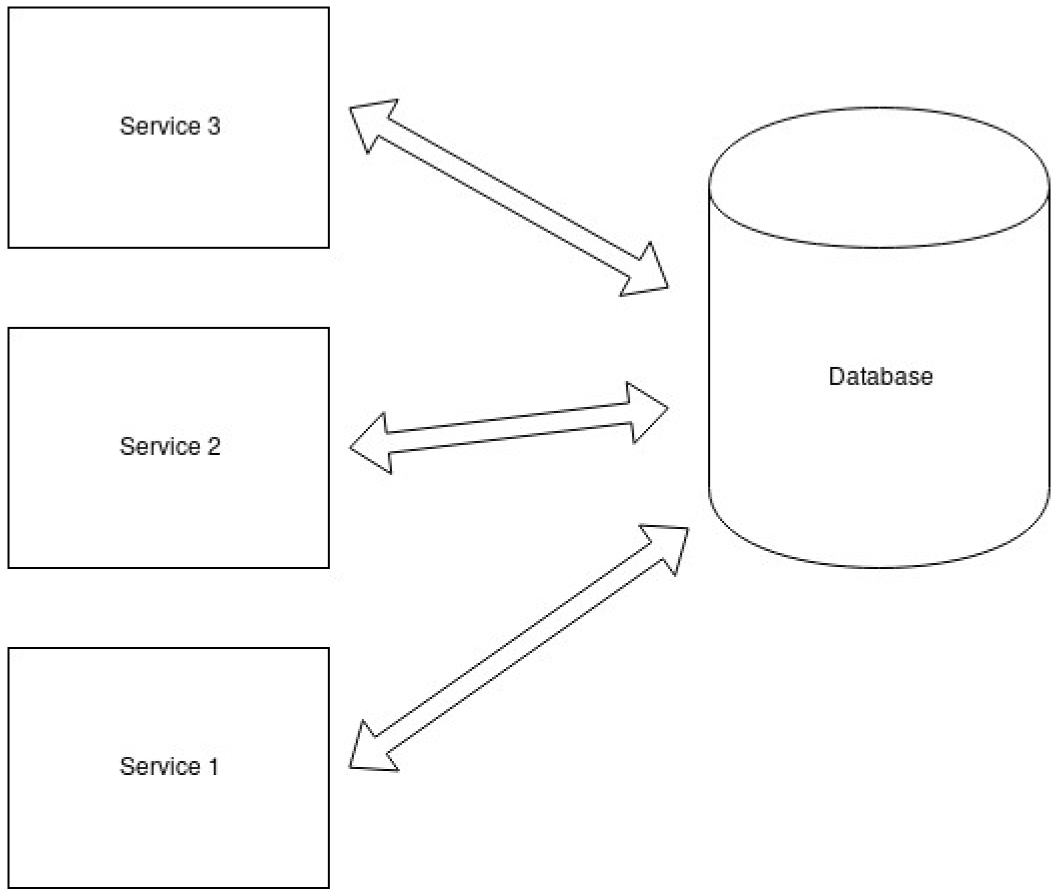
- Ruby programmer at Boostcom
- Blog: http://katafrakt.me
- Rails-free Ruby newsletter: https://rubytuesday.katafrakt.me
- NOT an Event Sourcing expert or theorist

EVENT SOURCING

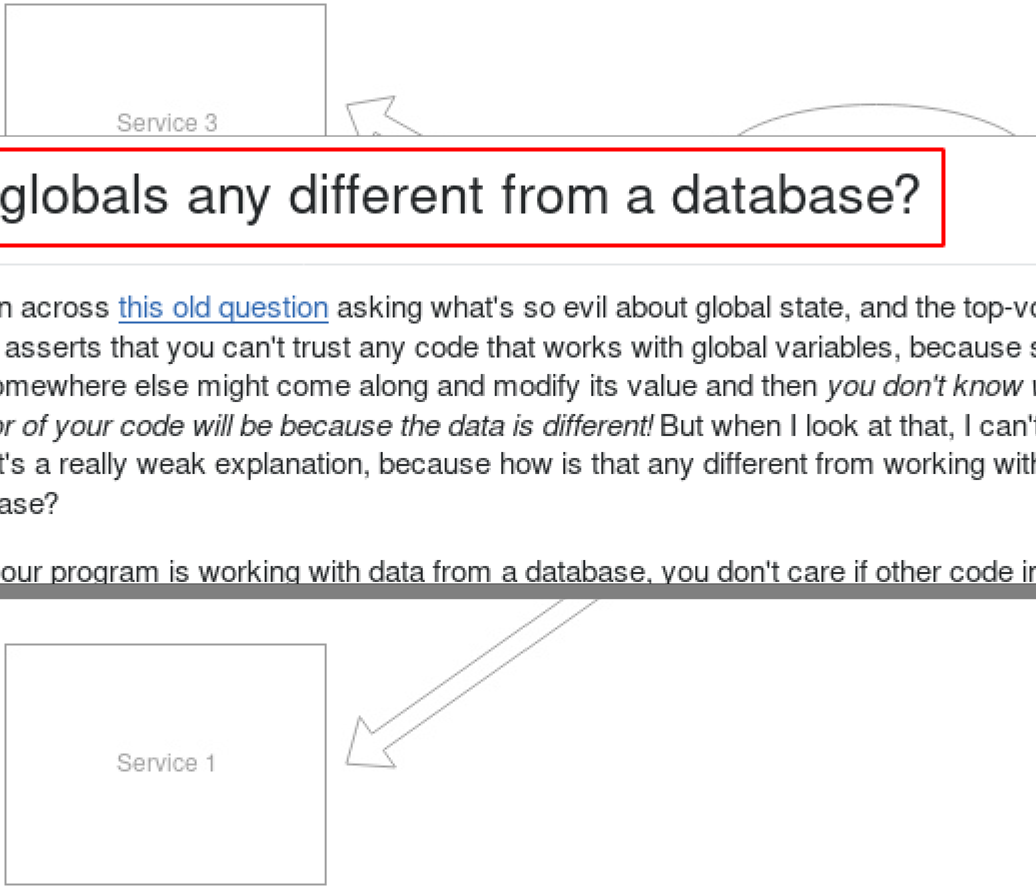
Typical web application

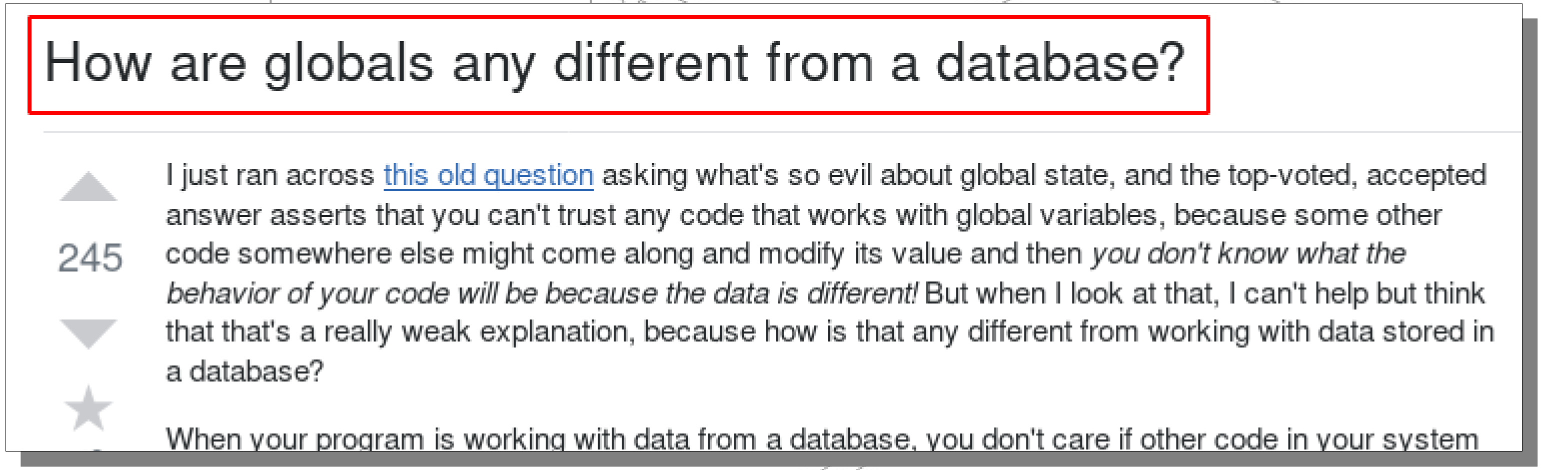


Typical modern web application



Typical modern web application




How are globals any different from a database?

245

I just ran across this old question asking what's so evil about global state, and the top-voted, accepted answer asserts that you can't trust any code that works with global variables, because some other code somewhere else might come along and modify its value and then *you don't know what the behavior of your code will be because the data is different!* But when I look at that, I can't help but think that that's a really weak explanation, because how is that any different from working with data stored in a database?

When your program is working with data from a database, you don't care if other code in your system

Typical modern web application as code

```ruby
$state = [{ id: 1, name: 'Element 1' }, { id: 2, name: 'Element 2' }]

def mutate!(i)
  $state.shuffle!
  $state[0][:name] = "Element #{i}"
end

5.times.map do |i|
  Thread.new { mutate!(i) }
end.each(&:join)
```

HOW DID WE END UP HERE?!?!?

- Logs
- Papertrail
- Audit log

What if we could have temporal data (history of changes) as first-class data?

Events

- `{ event: ‘name_change’, name: ‘Element 2’, id: 1, timestamp: 12345 }`
- `{ event: ‘name_change’, name: ‘Element 1’, id: 1, timestamp: 12346 }`
- `{ event: ‘name_change’, name: ‘Element 5’, id: 2, timestamp: 12347 }`
- `{ event: ‘name_change’, name: ‘Element 2’, id: 1, timestamp: 12348 }`
- `{ event: ‘shuffle_array’, ids: [2, 1], timestamp: 12349 }`

Events Sourcing rules

- Only append
- Always have timestamp
- Event source / event log is your single source of truth

Events Sourcing wins

- You always have the history
- History is first-class data, you can perform aggregations etc.
- In rare cases you can get system to the state of some time in the past and debug
- You always know where to seek the truth

Events Sourcing losses

- Harder to get current state
- Take up much more storage space
- Different mindset – harder to get people on board

Traditional tools for event store

PostgreSQL

EVENT STORE®

cassandra

mongoDB

elasticsearch

ElasticSearch – what it is?

- Built in 2010 as full-text search engine
- Widely used as E in ELK Stack
  - Elasticsearch – Logstash – Kibana
  - Centralized logging
- Basically a document store

ElasticSearch

Pros

- Can easily handle large amount of data
- Optimized for logs – append-only structure
- Good search and filtering capabilities
- Scripting in Groovy
- Kibana for free

Cons

- Complicated to set up properly
- Terrible query language
- Extensive scripting can kill the server

Average conference talk about Event Sourcing, Q&A part

Q: How to begin with ES?
A: Oh, just start gathering some events...

2 years later...

OK, now what?

HOW WE DO IT AT BOOSTCOM

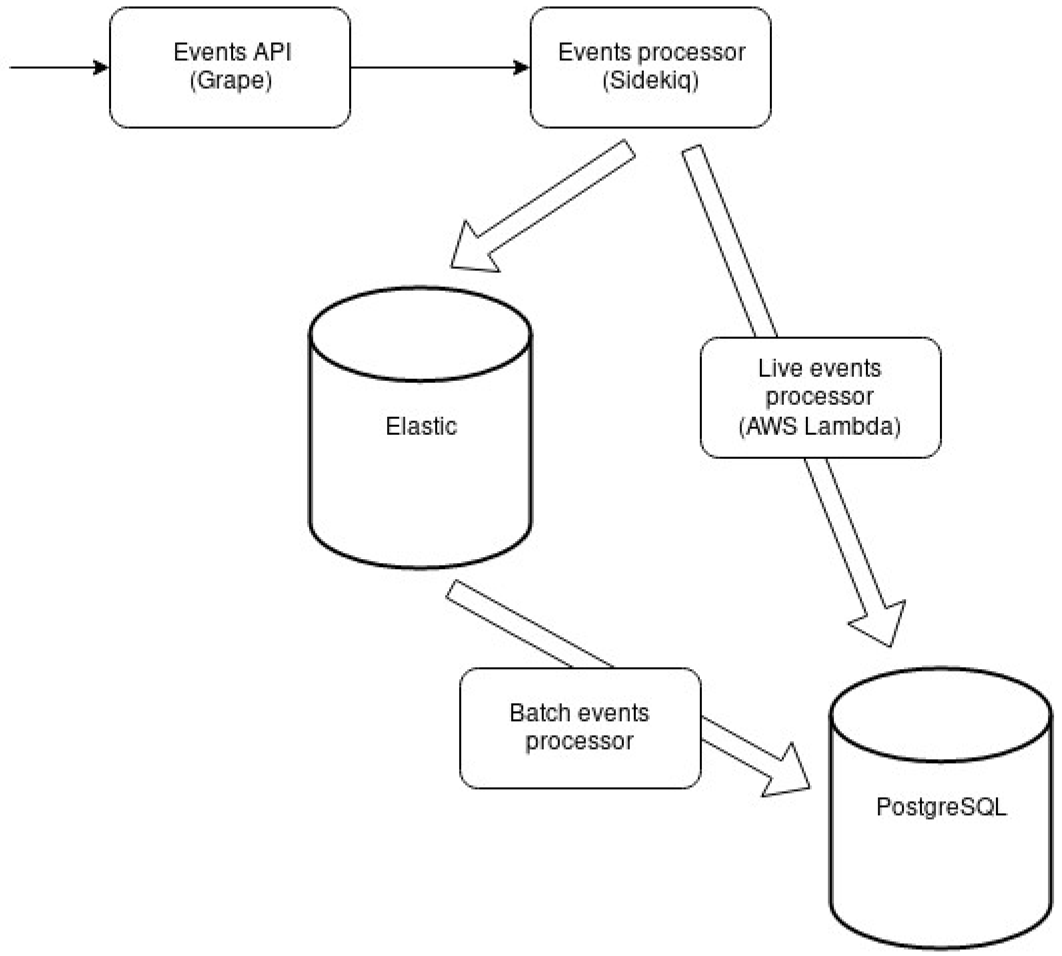
Putting it all together

Some facts

- Our traffic is mostly peak-based
- We handle 2 – 3 million daily (**2448995** – exact daily average from last two weeks)
- Some external integrations are really bad at resending events when something happens
- Others send duplicates on regular basis

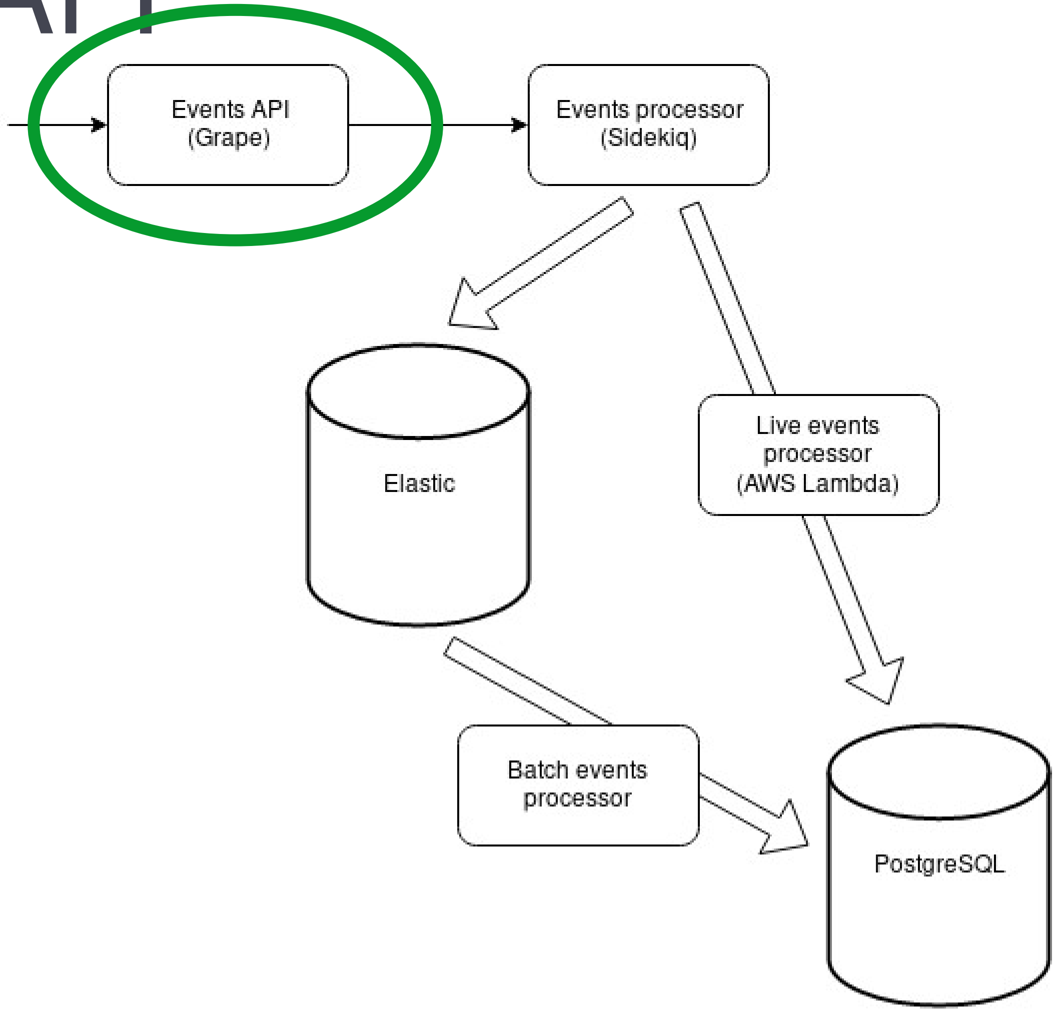
Architecture

- 4 components
- 2 databases
- 1 web application
- 1 serverless set of functions

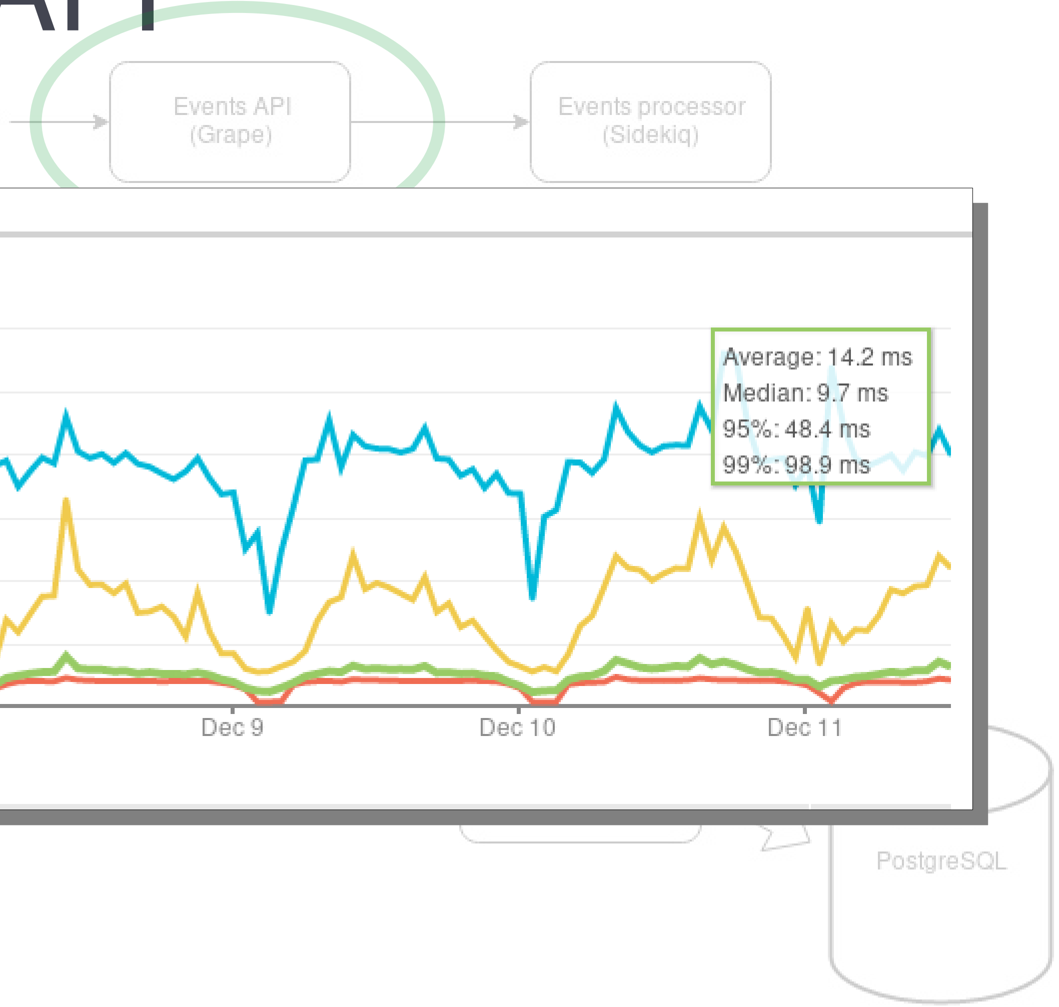

Events API (Grape)
Events processor (Sidekiq)
Elastic
Live events processor (AWS Lambda)
Batch events processor
PostgreSQL

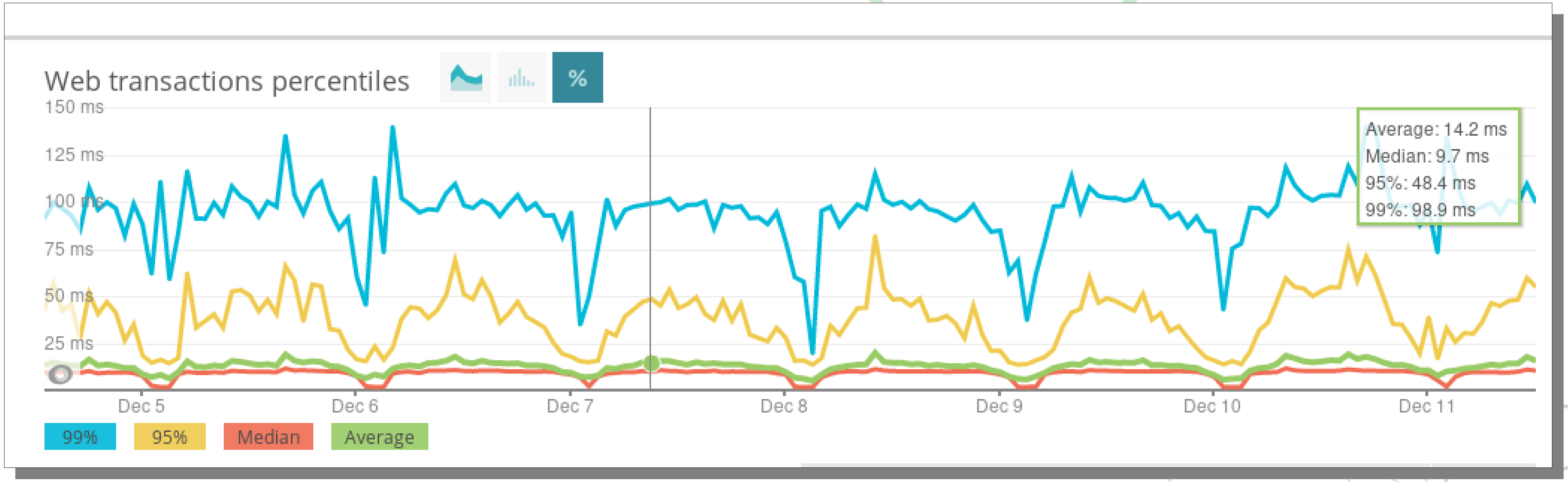
Architecture – Events API

- Only web-facing app
- HTTP API on Grape
- Only authorization and brief validation
    - If event happened, it happened
- Saves Sidekiq jobs to be picked up asynchronously
- Never failed ;)


Events API (Grape)
Events processor (Sidekiq)
Elastic
Live events processor (AWS Lambda)
Batch events processor
PostgreSQL

Architecture – Events API

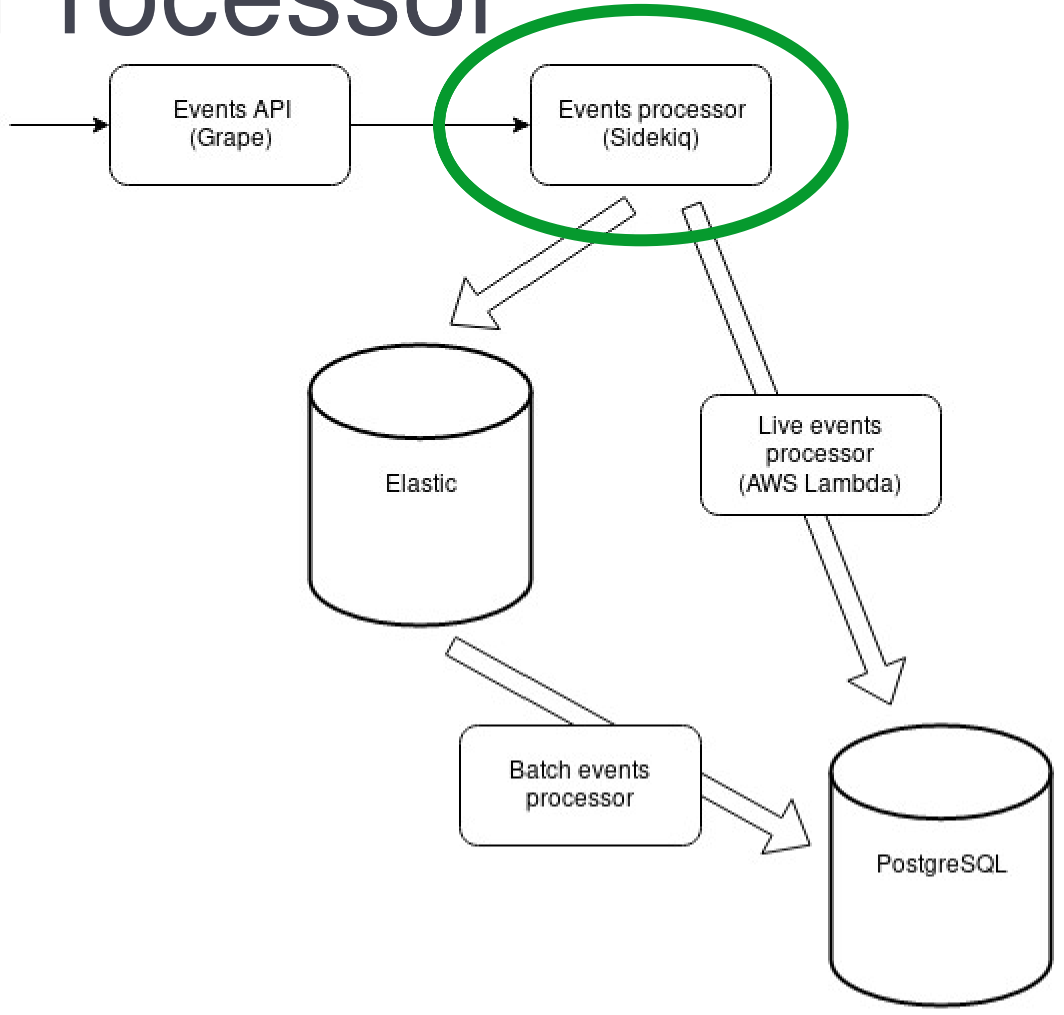




Average: 14.2 ms
Median: 9.7 ms
95%: 48.4 ms
99%: 98.9 ms

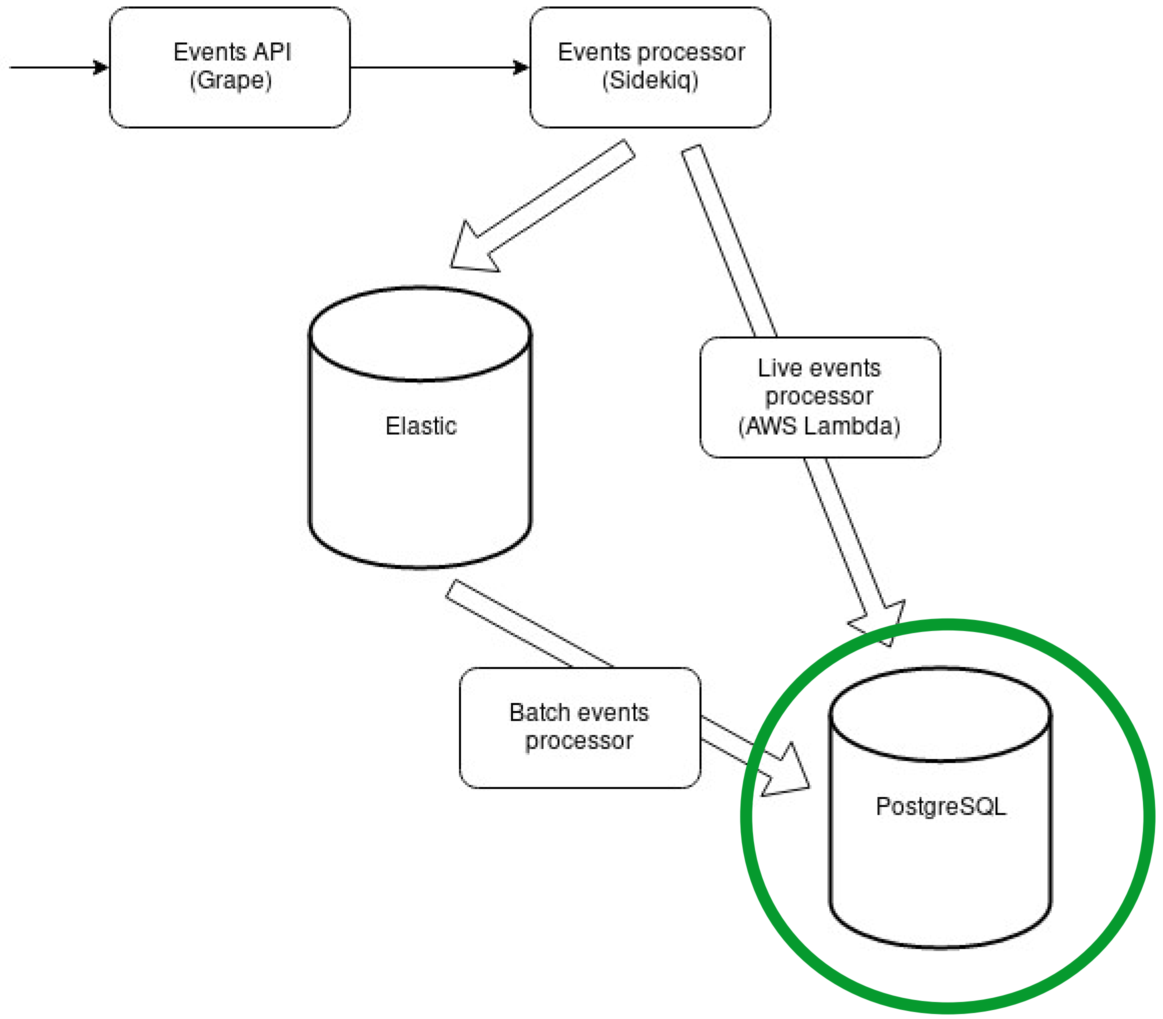
Architecture – Events Processor

- Does all the heavy lifting
- Checks for duplicates (checksums)
- Adds missing data
- Creates internal events
- Saves to ElasticSearch and Kinesis



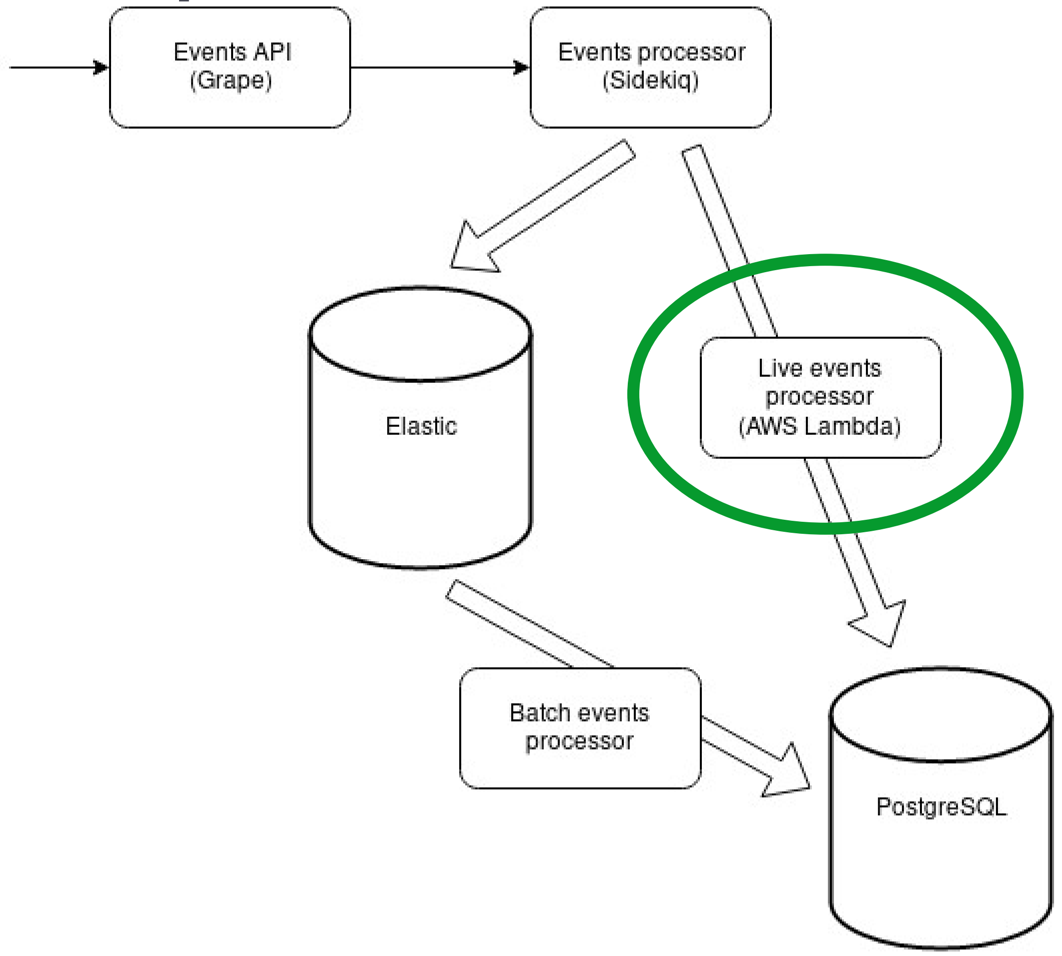
Architecture – read database

- Events log is not very well-suited for complex querying (no joins)
- We store data in read-optimized SQL database too


Events API (Grape)
Events processor (Sidekiq)
Elastic
Live events processor (AWS Lambda)
Batch events processor
PostgreSQL

Architecture – live events processor

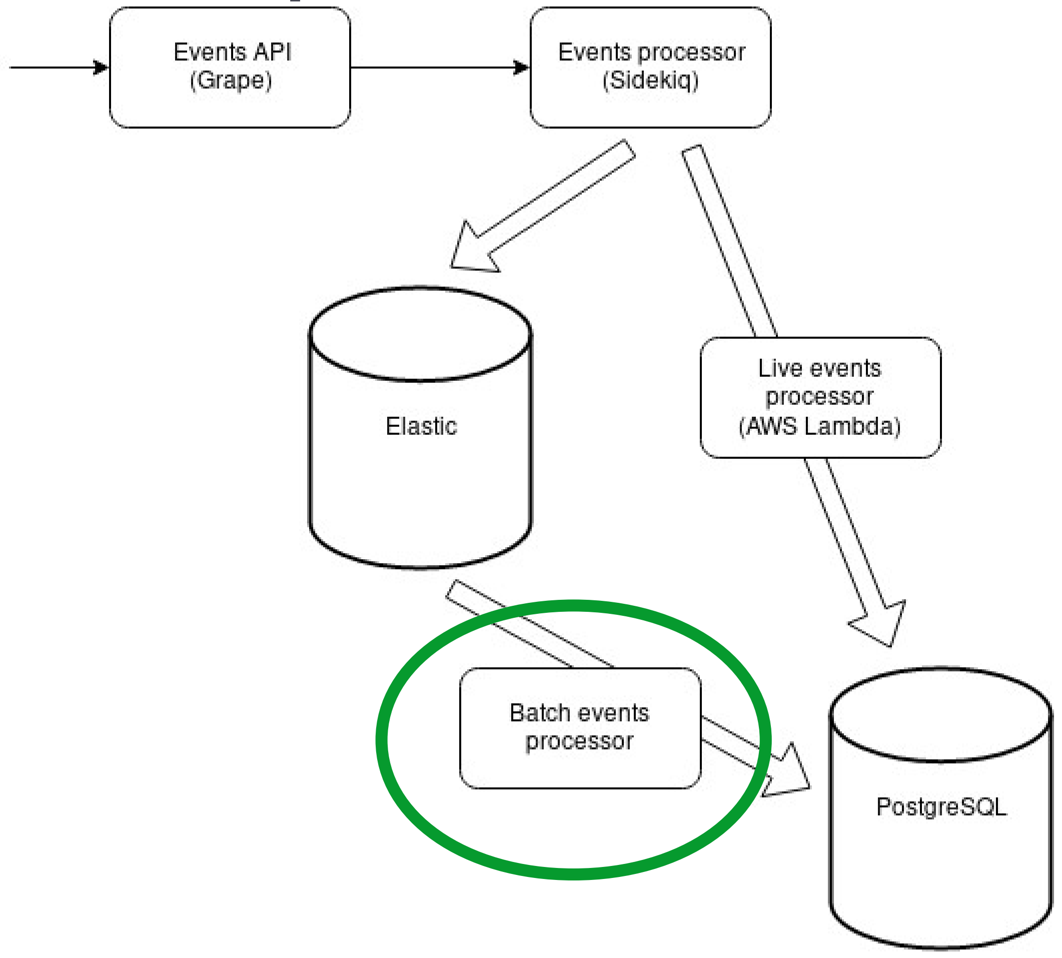
- Takes an event and "applies it" to read database
  (for example mark that user used at least one coupon)
- Written in Python :(



Events API (Grape)
Events processor (Sidekiq)
Elastic
Live events processor (AWS Lambda)
Batch events processor
PostgreSQL

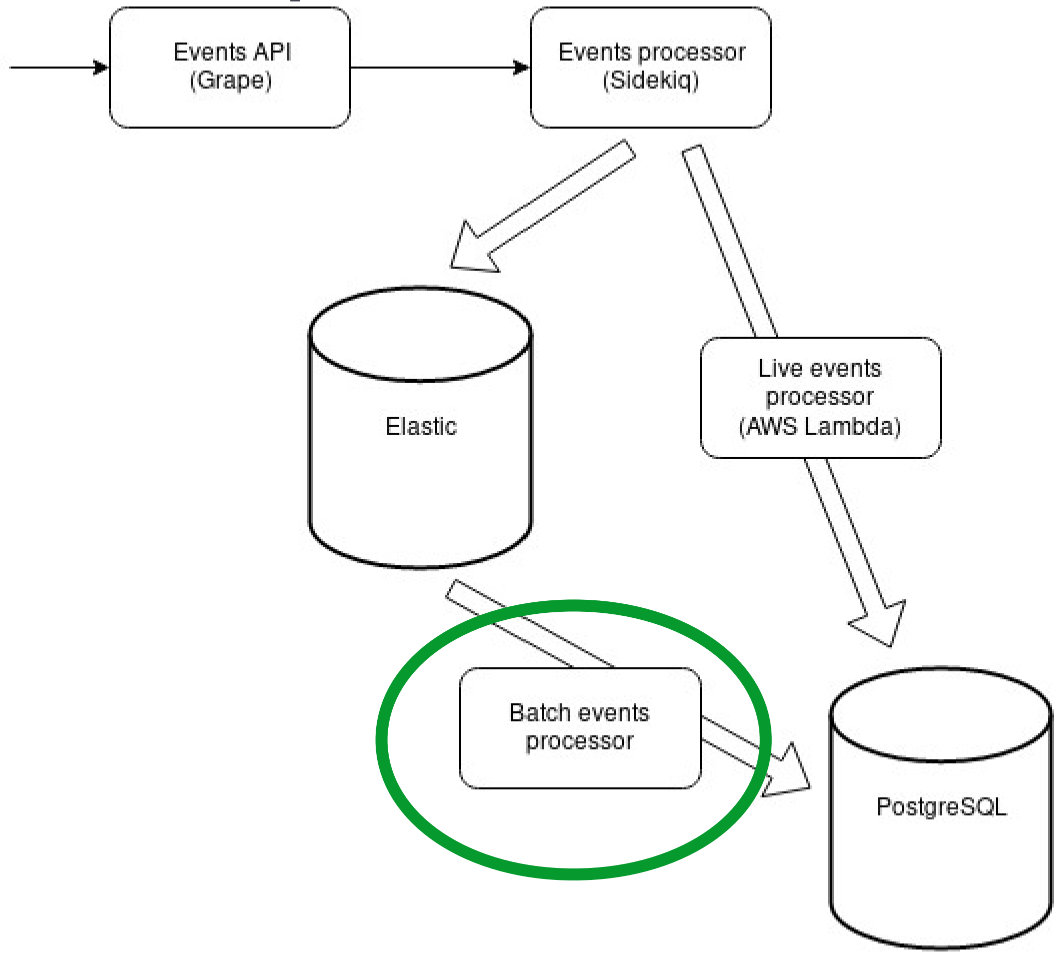
Architecture – batch events processor

- Drop read database and rebuild from scratch!
- Is not just applying events in order (would be too slow)
- I also relatively slow
- Some code duplicated from live processor



Architecture – batch events processor

- Written in Ruby
- Uses RocksDB for storing intermediate results
- Outputs CSVs which are COPY’d to PostgreSQL
- Uses ElasticSearch scrolls

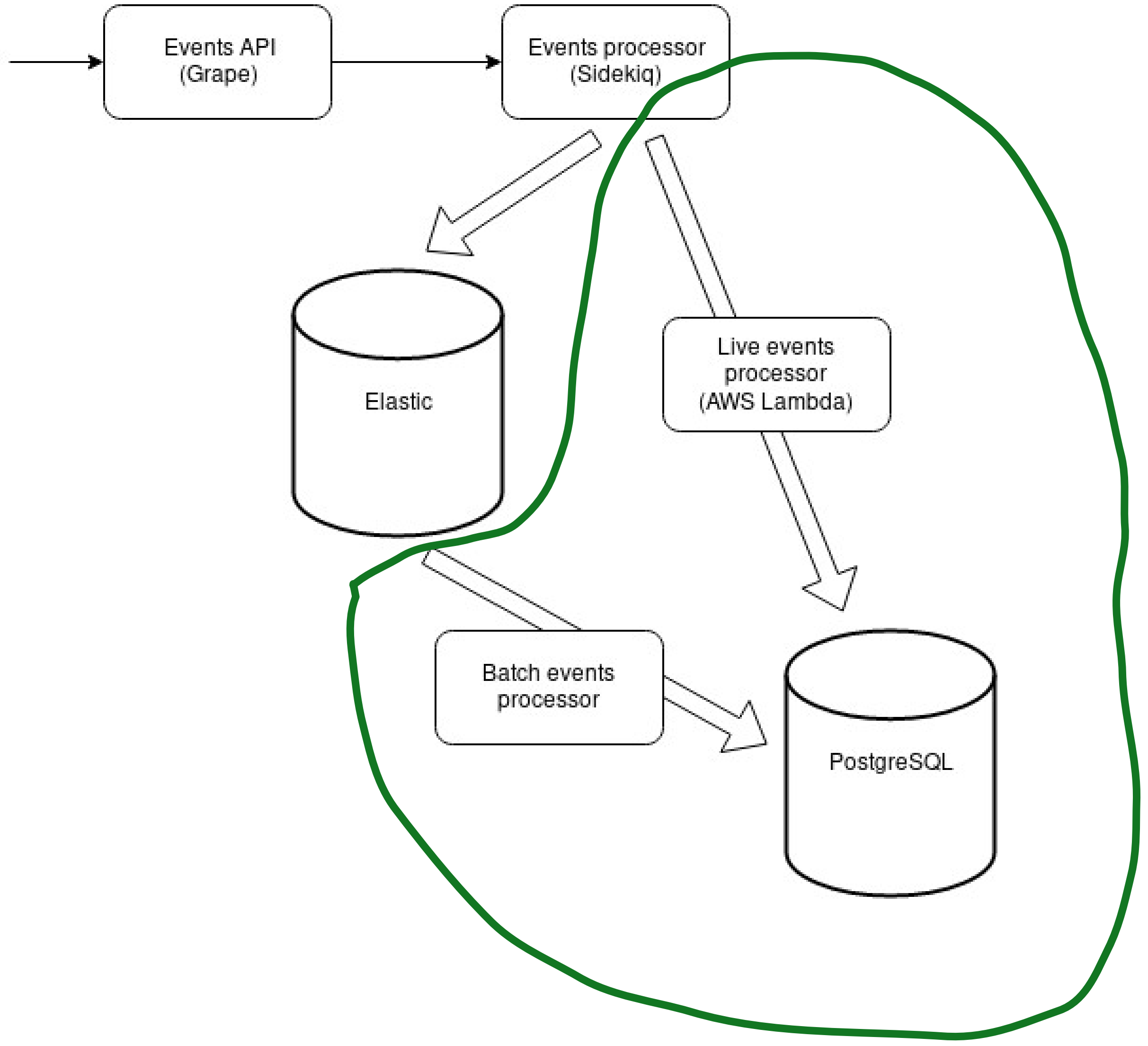


Part where you can make mistakes

- Wrong database layout or missing column?
- Some events processed in wrong order?
- Database or Lambda down?

NO PROBLEM*

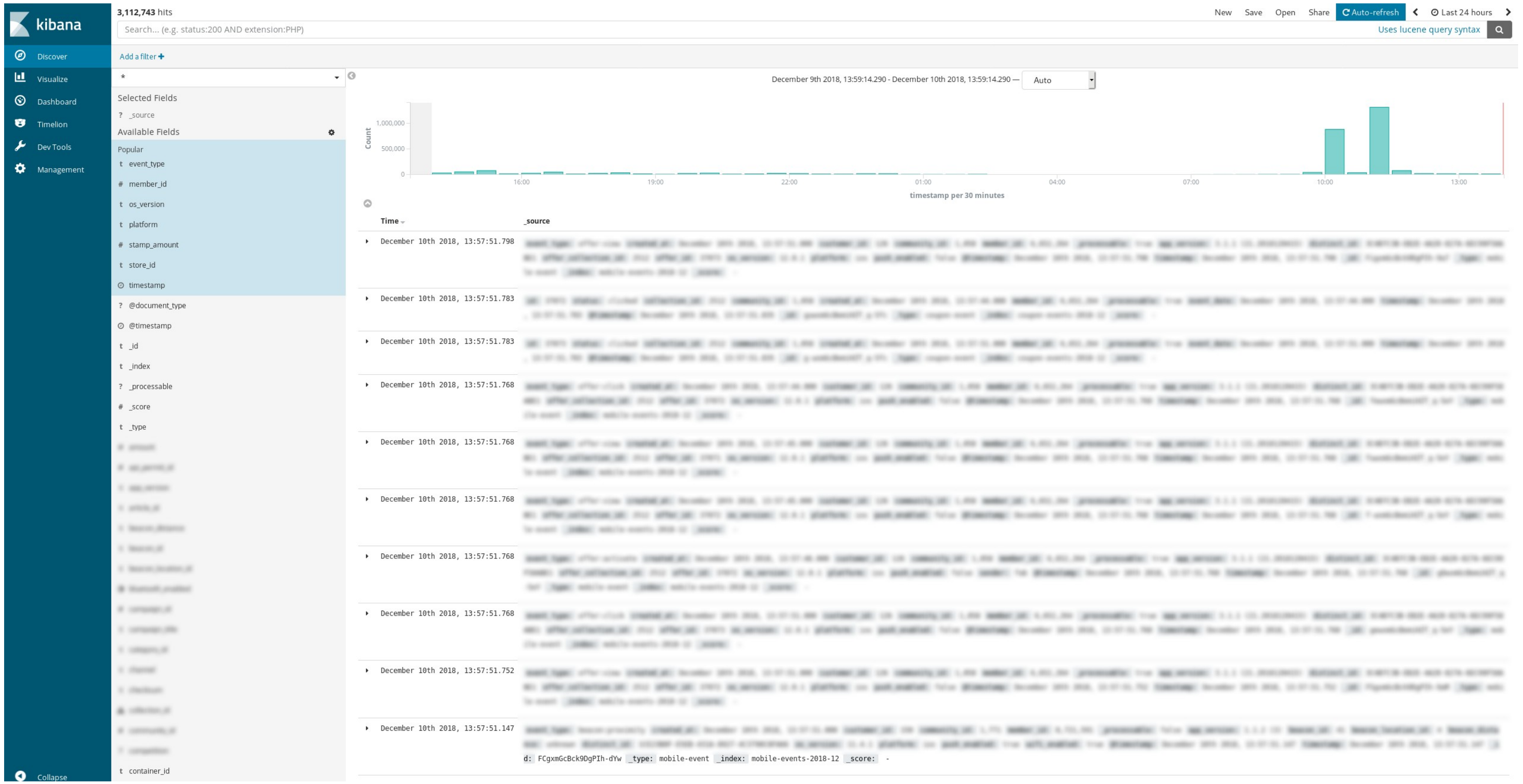
(* You're not losing data)


Events API (Grape)
Events processor (Sidekiq)
Elastic
Live events processor (AWS Lambda)
Batch events processor
PostgreSQL

Benefits of ElasticSearch

- Events in store are append only and immutable, but…
    - ES allows updates without rewriting the whole log (can be done live)
- We are not afraid about growing data volume
- Can connect to many existing solutions (PowerBI)
- Basic insights using Kibana

Kibana – event filtering, viewing etc.



3,112,743 hits

Search... (e.g. status:200 AND extension:PHP)

*

Selected Fields

? _source

Available Fields

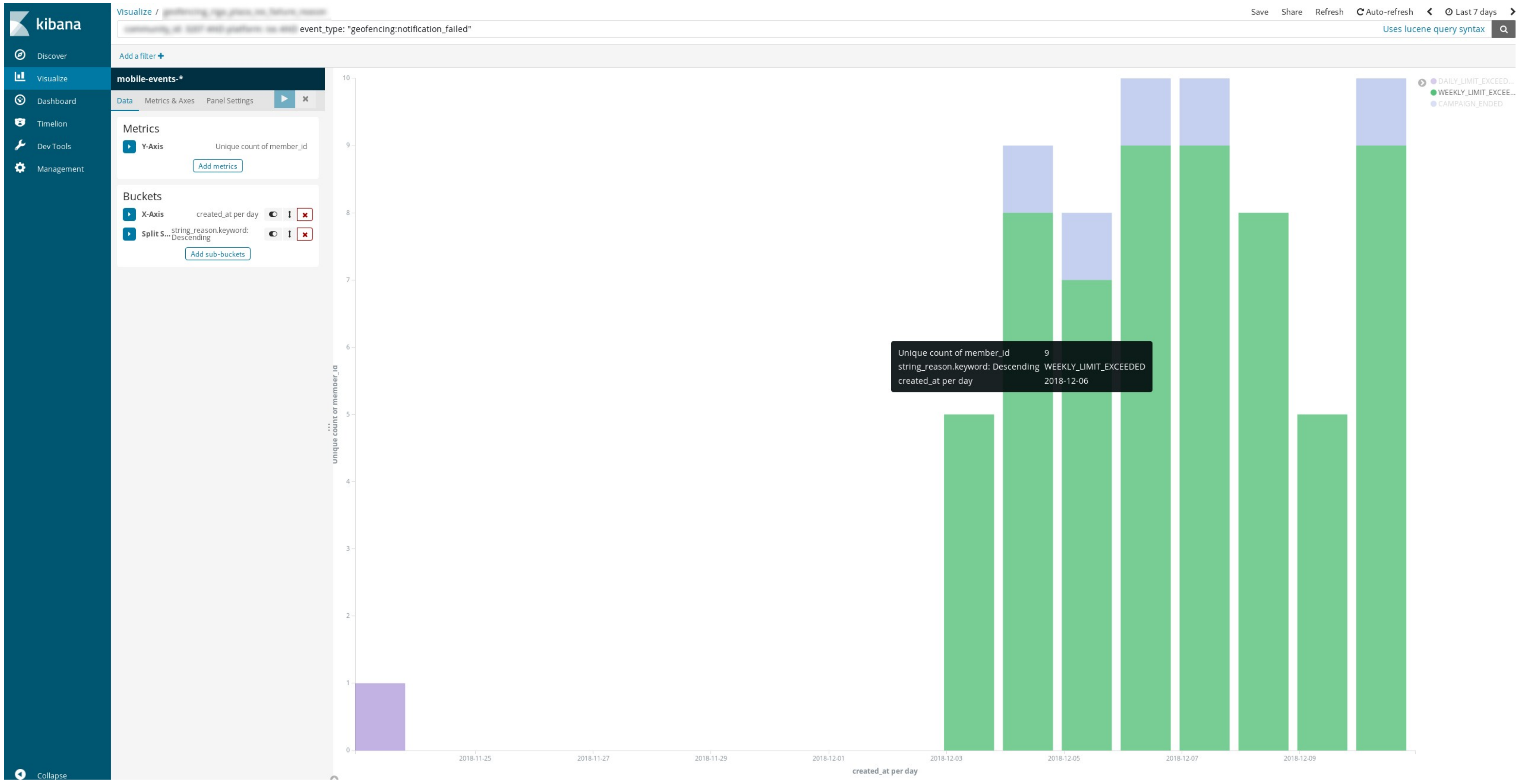
Popular

- t event_type
- # member_id
- t os_version
- t platform
- # stamp_amount
- t store_id
- timestamp
- ? @document_type
- @timestamp
- t _id
- t _index
- ? _processable
- # _score
- t _type
- [redacted]
- t container_id

December 9th 2018, 13:59:14.290 - December 10th 2018, 13:59:14.290 —

Time	_source
December 10th 2018, 13:57:51.798	[redacted]
December 10th 2018, 13:57:51.783	[redacted]
December 10th 2018, 13:57:51.783	[redacted]
December 10th 2018, 13:57:51.768	[redacted]
December 10th 2018, 13:57:51.768	[redacted]
December 10th 2018, 13:57:51.768	[redacted]
December 10th 2018, 13:57:51.768	[redacted]
December 10th 2018, 13:57:51.768	[redacted]
December 10th 2018, 13:57:51.752	[redacted]
December 10th 2018, 13:57:51.147	[redacted] d: FCgxmGcBck9DgPIh-dYw _type: mobile-event _index: mobile-events-2018-12 _score: -

Kibana – visualizations



Visualize / [redacted]

[redacted] event_type: "geofencing:notification_failed"

mobile-events-*

Metrics
Y-Axis Unique count of member_id

Buckets
X-Axis created_at per day

Split S... string_reason.keyword: Descending

```
Unique count of member_id              9
string_reason.keyword: Descending      WEEKLY_LIMIT_EXCEEDED
created_at per day                     2018-12-06
```

Kibana benefits

- We were able to teach some non-programmers to use Kibana before asking questions
- It’s faster for us too to use it’s interface for checking things
- You cannot break anything

I TAK SIĘ POWOLI ŻYJE
NA TEJ WSI
Memy.pl

Tips & Tricks

- Don’t try to event source everything. Because:
  - tech constraints (synchronous things, uniqueness checks…)
  - not real events
  - law (GDPR *et consortes*) - remember about personal data!
- Collect many events, not only ones you think you need now
- Don’t be afraid to give up on some events if their quality is bad
- Consider seeding

BOOSTCOM

WE’RE HIRING

¿QUESTIONS?