Elixir is not Ruby

#ELFBA 2017 - Buenos Aires 🇦🇷

$ whoami

- Thomas Gautier 🇫🇷
- Co Founder & CTO at fewlines
- We don't have a website but we maintain an Open Source package: bamboo_smtp
- Teacher, Architect, Developer, DevOps
- API Lover
- Former teacher at *Le Wagon Paris & Lille* (9 weeks Ruby bootcamp)

Lille 🇪🇺


AMSTERDAM
2h40
LONDON
1h30
Calais
Dunkerque
Boulogne
BRUSSELS
35 min
KÖLN
2h35
LILLE
3h
LUXEMBOURG
Roissy CDG
50 min
PARIS
1h


TERKEN
Orland
CH'TI
Blanche
CH'TI
La grande de l'Aunette
ENLAIN
OULINS D'ASCQ
66
SECRET
La Goudale
3 MONTS
Gavroche


nnais un homme
rrêté
e

$ man fewlines

- We provide Tech Education 🤓
- We build ⚡ **API**-first Software ⚡ for the Retail industry
- We 😍 **Elixir**!

Community 😍

- **LilleFP**:
  - Biggest french speaking Meetup on Functional Programming
  - ~50 attendees
  - Every 2 months
- **Elixir |> Lille**
  - One of the "biggest" french speaking Meetup
  - ~30 attendees
  - Quarterly

Why this talk?

- We're developers first,
- but we're also teachers,
- to beginners and seasoned developers alike.

and we used to do it

with Ruby

WARNING !

DISCLAIMER

We ❤️ Ruby

We just don't use it anymore

but there's a problem

Follow

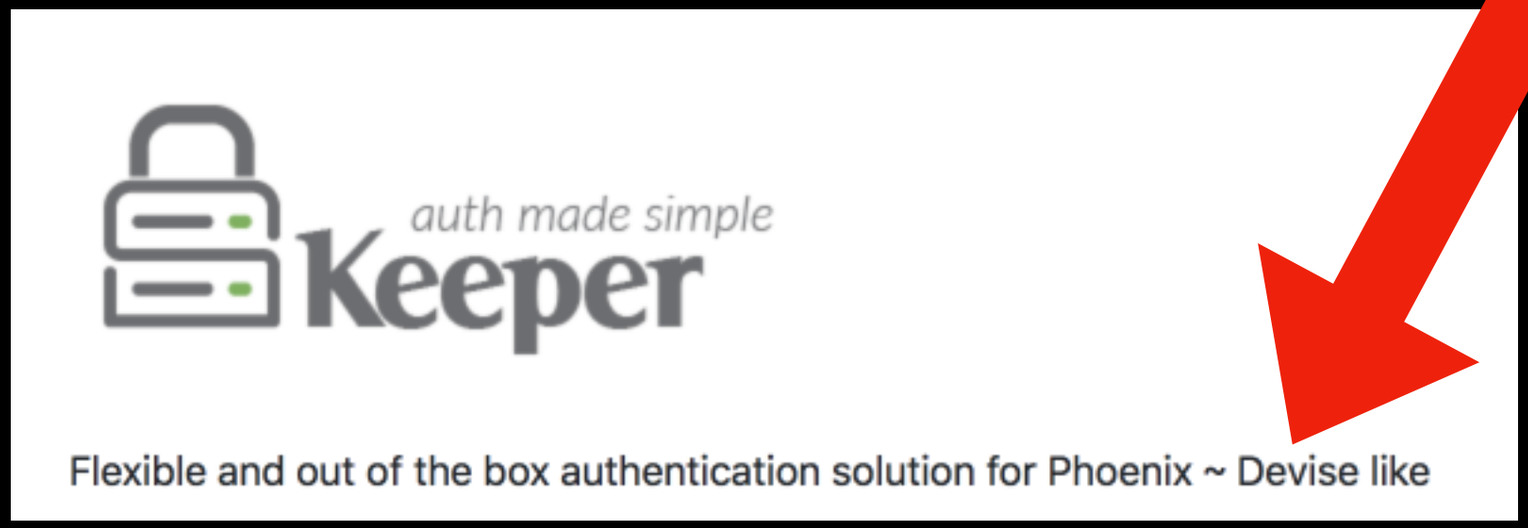
"Phoenix and Elixir are not a paradigm shift in how we program, but it is a beautiful evolution of Rails." via @thoughtbot #myelixirstatus

WAT!

Retweets Likes

3 6 9

WAT!


auth made simple
Keeper
Flexible and out of the box authentication solution for Phoenix ~ Devise like

and this makes us go

Because the only thing that Ruby and Elixir have in common

is this guy

José Valim - Elixir creator, former Rails core team.

From OO to FP

This is our journey, as developers, and as teachers

Elixir is not Ruby

because it's obviously
not Object Oriented

Even if Processes & Messages
may be the "*best*" implementation
of Alan Kay's ideas

Elixir

- Modules
- Functions as 1st class citizen
- Immutability

Elixir ❤️

- Pattern Matching
- Protocols (Java Interfaces 👨🏾‍🔬)
- Macros (LISP baby 👩🏽‍🔬)

Thanks Captain Obvious 👍🏾

Elixir is Functional

so we focused on

Functional

and learned about

- Function as the primary (if not only) abstraction
- Immutability of data (although we can rebind variables)
- Expressions over Statements (we all love return values)
- Referential Transparency (same output for same input)

This helped a lot

But for the lower level abstraction only

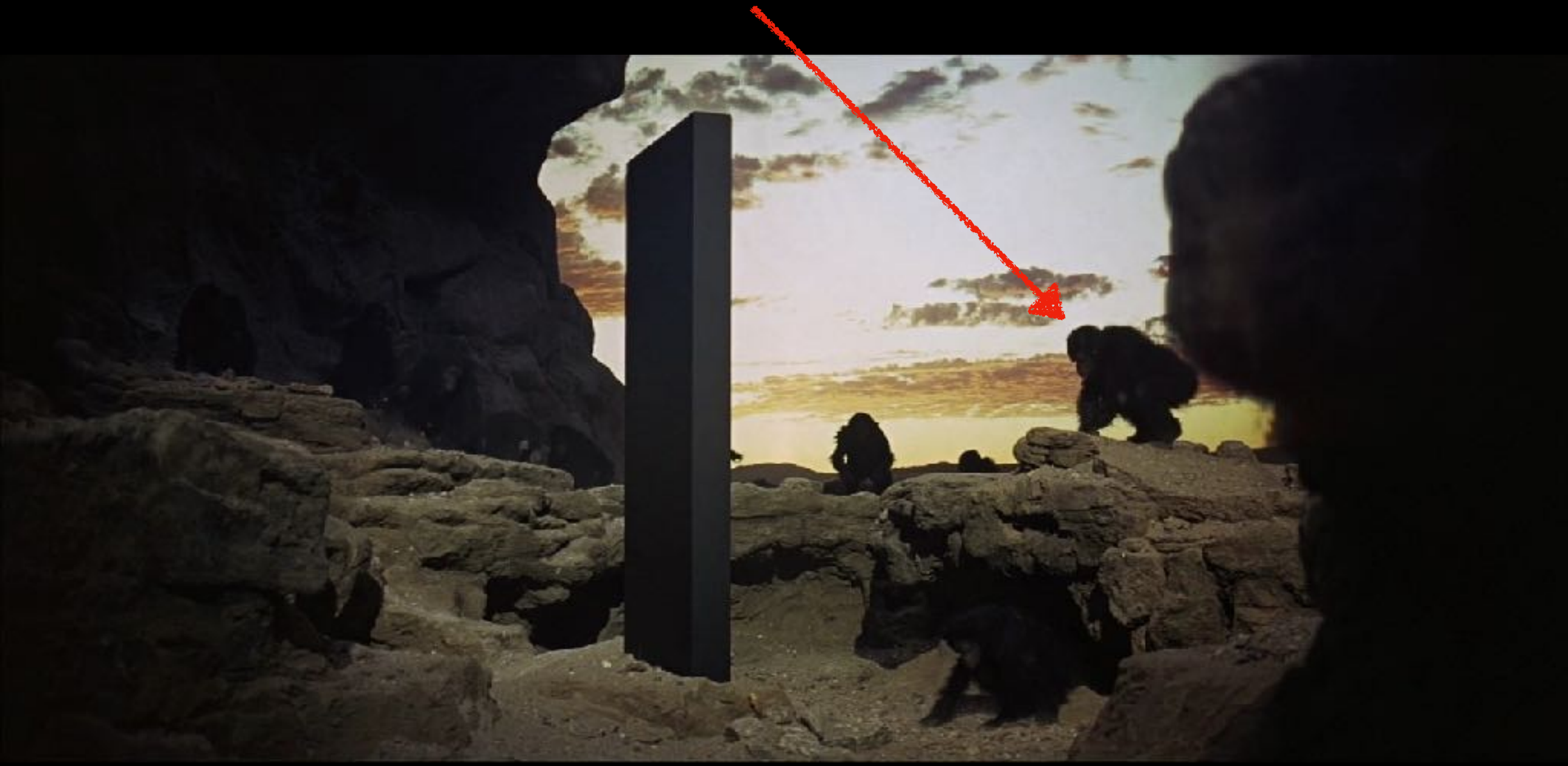
Because we were using Phoenix*

* pre 1.3, since then Chris McCord's ideas are really going in the right direction 🚀

So, except for Ecto

At the upmost level of abstraction, our app looked like a good old Rails monolith

This is 😞 me



- We did everything by the book(s) (and we did read a lot of them)
- But we felt like just doing plain old Model/View/Controller stuff was not getting us anywhere
- Even if we could call our code *Functional*

So yes, Elixir is

Functional

But it didn't make us feel any better

Because we heard about OTP promises

and we didn't feel
those superpowers

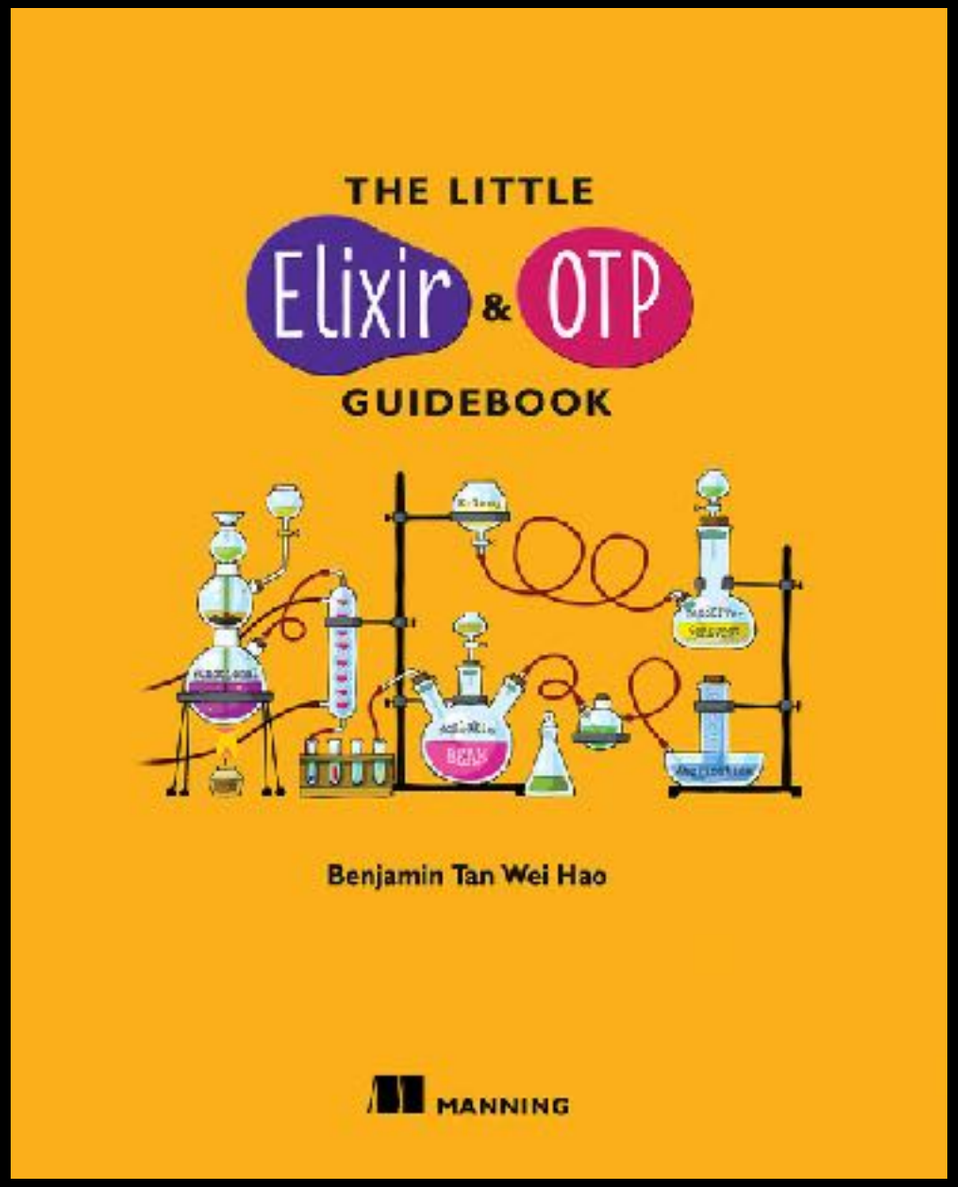
- Concurrency
- Distribution
- Fault Tolerance
- (D)ETS & Mnesia
- Hot Code Reload & Releases
- Cookies 🍪 and Poneys 🦄

So Elixir is not (only)
about being Functional

But it's (mostly) about OTP patterns

So we looked for a book

and we found one!



The Little Elixir & OTP Guidebook

Benjamin Tan Wei Hao

Especially the parts on

- Processes
- Monitors
- Supervisors
- Implementing a Workers Pool Supervisor (this was hard 😰)
- Re-implementing GenServer (a naïve one, we're not that good)

So we got the
Concurrency part

But what about Distribution?

Well, it kinda worked

but just on one node...

How do we distribute state across nodes?

José says we don't really need Distribution

but we ***do*** need it...

We have lots (millions) of users, all around the world

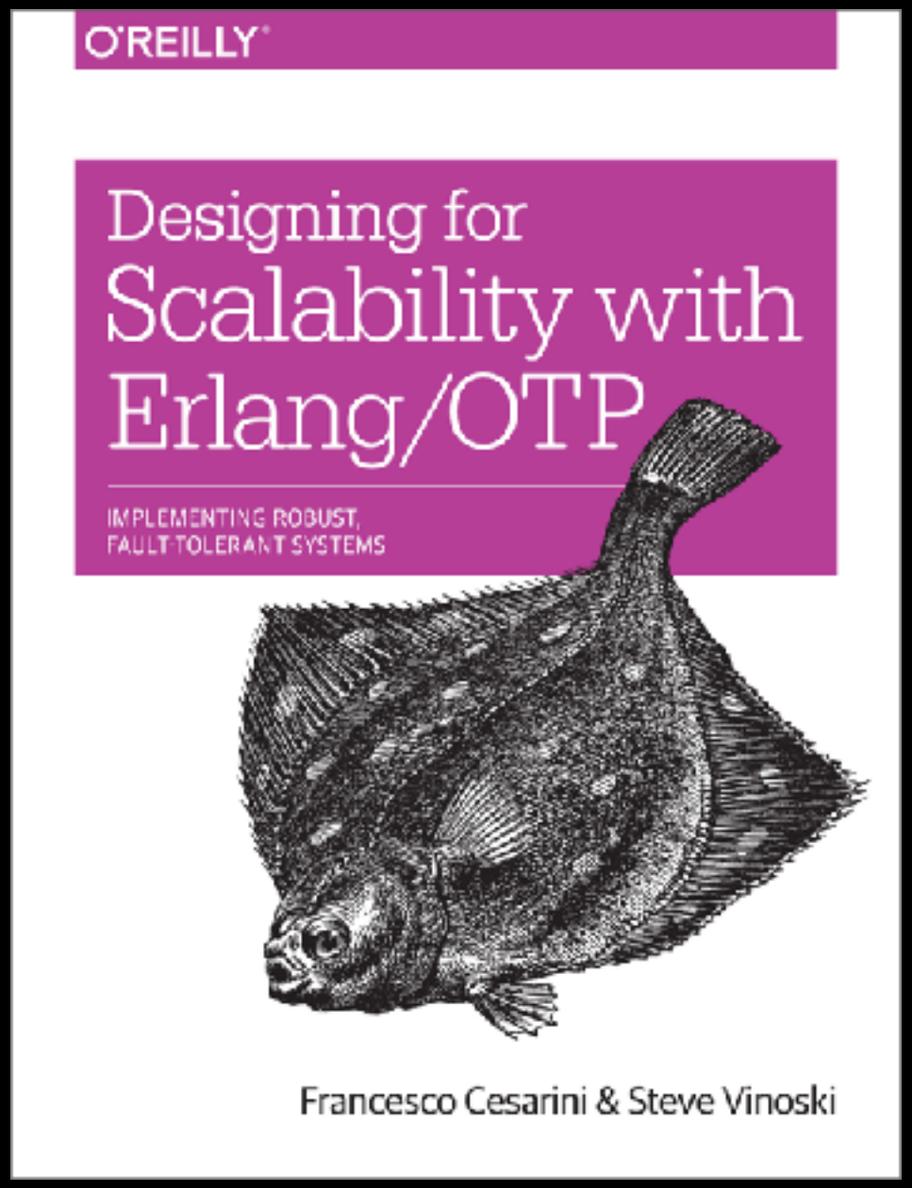
with a real need of
online/offline strategies

And this is where we felt kind of lost...

Not that many articles available online

And not a lot of
books either

except one



Designing for Scalability with Erlang/OTP

Francesco Cesarini and Steve Vinoski

Thanks Francesco for finally finishing your book 🤗

So did we find a solution for Distribution?

No, we didn't find one...

but we've found 3 of them 😊

1. OTP Only

- No dependency (BEAM as your OS 😈)
- Fully meshed Nodes topology
- Every node is named, based on the "application" it executes:
  - `business_2`,
  - `cache_4`,
  - `proxy_42`
- Then we know which nodes to contact with a random call

2. Cloud Cheat

- Rely on AWS tag for instances (ie: `cache`, `business`, `proxy`)
- This could work with any Cloud Provider that supports it
- At Node startup, it reads the AWS tags and registers itself to a custom Registry (duplicated through the cluster)
- Each time we need to do a call to a GenServer, pick a random `tag`, whatever the Node and call it
- If we receive a `nodedown` message, remove the Node from the Registry

3. RabbitMQ 🥕

- New dependency in the Stack 😟
- Push data to RabbitMQ
- Use GenStage to handle incoming data from RabbitMQ

We don't know which one will fit us the best

But we'd love to discuss about it with you

Conclusion

- Elixir, Erlang & OTP are amazing platforms with great communities
- Avoid the Ruby and Elixir analogies
- Insist on the unique advantages of OTP (Concurrency, Distribution, Supervision) instead of Functional idioms
- Try to (re)implement by yourself GenServer, Supervision Trees, etc... This is how we really got into OTP
- Share more on Distribution, this is the killer feature of OTP

Thanks 🙇

Thomas Gautier - @Aryko

fewlines

blog.fewlines.co