WE'RE ALREADY DOING MICROSERVICES!

CONTENTS

- INSTAMIKE PAINS 😫...
- BOUNDED CONTEXTS 
- WHAT ARE THESE MICRO THINGS 👾?
- 👾 GOTCHAS 😛
- MUST WIN.

BASED ON A

TRUE STORY

INSTAMIKE  

BASED ON A

TRUE STORY

MUSTWIN 

WHAT'S A RAILS MONOLITH?

- AN APP THAT'S SO BIG AND COMPLEX THAT IT'S SLOWING DOWN YOUR TEAM
- IT'S A SINGLE POINT OF FAILURE
- IT'S COMPLEXITY MAKES SIMPLE TASKS PAINFUL

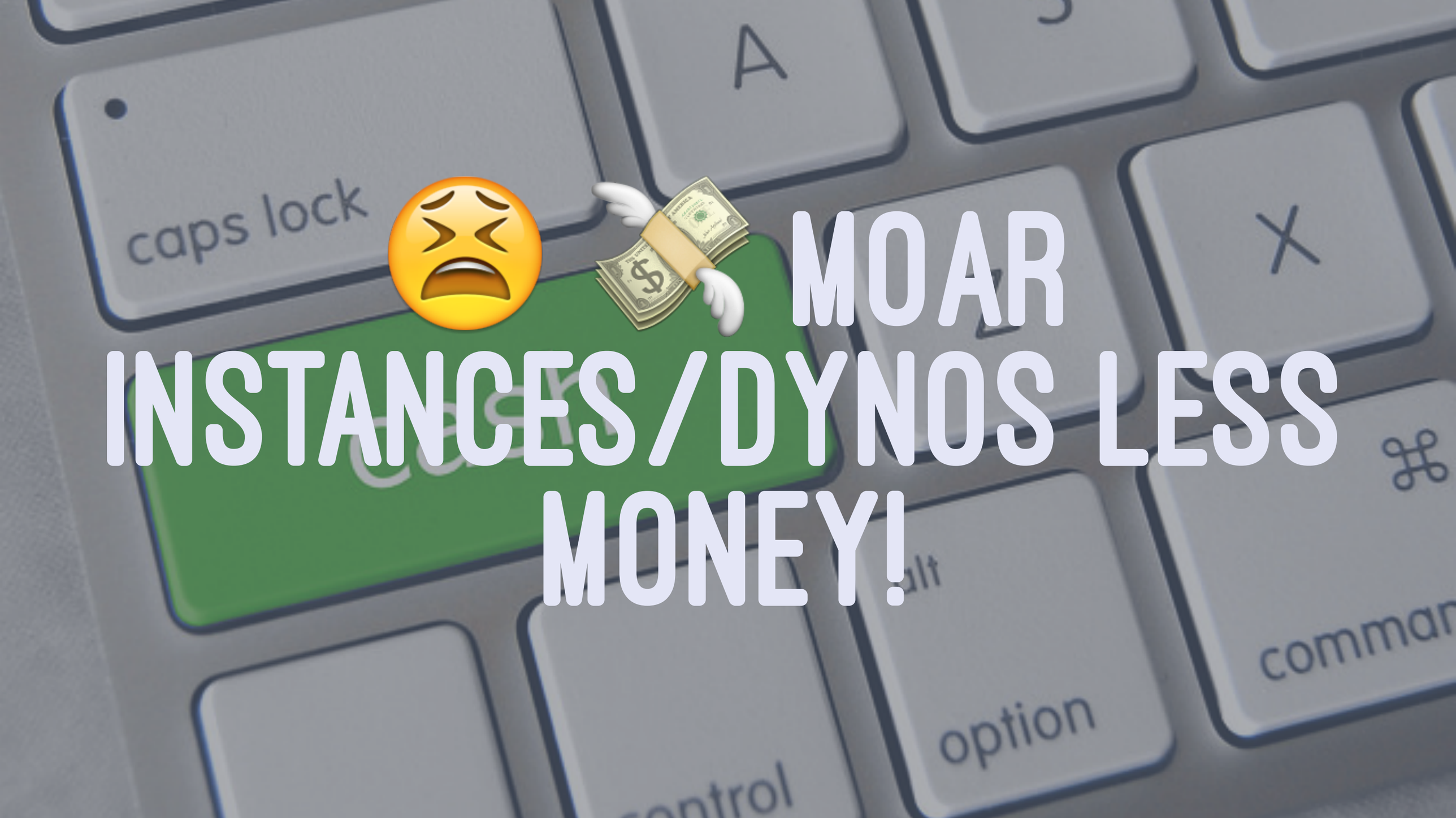
😫 GEM UPDATES WERE PAINFUL

^FOR INSTAMIKE. ONE OF THOSE  THINGS THAT BECAME MORE AND MORE PAINFUL AS
THE APP GREW WAS UPDATING GEMS.

😫 GEM UPDATES WERE PAINFUL

```
Bundler could not find compatible versions for gem "tilt":
  In Gemfile:
    sass-rails (= 3.2.6) ruby depends on
      tilt (~> 1.3) ruby

    slim (>= 0) ruby depends on
      tilt (2.0.0)
```



MOAR INSTANCES/DYNOS LESS MONEY!

😫💸 MOAR INSTANCES FOR JUST ONE ENDPOINT

- INSTAMIKE HAD A USER/IMAGES ROUTE IN THEIR APP. IT ACCOUNTED FOR 98% OF IT'S TRAFFIC.
- THEY KEPT HAVING TO BUY MORE BOXES JUST FOR THIS ONE ENDPOINT.
- IT WAS 💰💰💸


THE ENGINEERS MISSED
SATURDAYS 😫 😫
this weekend take the train,

SATURDAY WAS THE ONLY "SAFE DAY" TO DEPLOY 😫

- IF EVERYTHING LIVES INSIDE ONE APP, THEN THE WHOLE TEAM NEEDS TO BE IN SYNC WHILE DOING RELEASES.
- FOR INSTAMIKE THAT MEANT THAT ALL RELEASES HAD TO HAPPEN ON SATURDAY MORNING WHEN THE USERS WERE LESS ACTIVE.
- IT WAS 😫

HOW DID THEY FIX INSTAMIKE?  DIVIDE AND CONQUER!

1. IDENTIFY WHAT THINGS BELONG TOGETHER
2. ISOLATE THE PARTS
3. SCALE THOSE PARTS

BOUNDED CONTEXTS FTW! 

"A SPECIFIC RESPONSIBILITY ENFORCED BY EXPLICIT BOUNDARIES"

A SMALL SET OF OF MODELS THAT OPERATE ALMOST INDEPENDENTLY OF THE REST OF THE APP.

BOUNDED CONTEXTS FTW! 

```
$ tree libs/
libs/
├── analytics
│   ├── ab_groups.rb
│   └── generate_report.rb
├── photos
│   ├── filters.rb
│   ├── resize.rb
│   └── upload.rb
└── user
    ├── friends.rb
    └── user.rb
```

BOUNDED CONTEXTS FTW! 

- RAILS ENGINES ARE BOUNDED CONTEXTS

```
module Blorgh
  class Engine < ::Rails::Engine
  end
end
```

DEVISE CONTEXT -> AUTH-ER

- DEVISE IS THE MOST USED AUTHENTICATION GEM IN RAILS
- IT'S AN ENGINE THAT ALLOWS YOUR USERS TO BE LOGGED IN OR OUT

DEVISE CONTEXT -> AUTH-ER

- IN A NUTSHELL WHAT DOES DEVISE DO? SETUP A WARDEN COOKIE
- THAT'S IT
- YOUR APP IS RESPONSIBLE FOR WHAT TO DO WITH THE "LOGGED USER"

DEVISE CONTEXT -> AUTH-ER

- COULD YOU SET THAT COOKIE ON ONE RAILS APP AND USE IT IN ANOTHER?
  - YES!

THE AUTH-ER APP IS NOW A SIMPLE DEVISE APP WITH JUST THESE THREE GEMS

```
source 'https://rubygems.org'
```

DEVISE CONTEXT -> AUTH-ER

- AND YOU CAN EVEN USE THESE SESSIONS OUTSIDE OF RAILS
  - HTTPS://GITHUB.COM/MATTETTI/GORAILSYOURSELF

```
railsSecret := "f7b5763636f4c1f3ff4bd444eacccca295d87b990cc104124017ad70550edcfd22b8e89465338254e0b608592a9aac29025440bfd9ce53579835ba06a86f85f9"
encryptedCookieSalt := []byte("encrypted cookie")
encryptedSignedCookieSalt := []byte("signed encrypted cookie")

kg := KeyGenerator{Secret: railsSecret}
secret := kg.CacheGenerate(encryptedCookieSalt, 32)
signSecret := kg.CacheGenerate(encryptedSignedCookieSalt, 64)
e := MessageEncryptor{Key: secret, SignKey: signSecret}
```

THE IMAGE CONTEXT

ON MANY SOCIAL APPS IMAGES NEED TO BE:

* RESIZED
* SCALED
* CACHED
* UPLOADED

 -> 👾 IMAG-ER

CAN BE IMPLEMENTED WITH JUST 3 LINES:

```
$ gem install magickly
$ gem install thin
$ thin start
```

HTTP://
IMAGER.INSTAMIKE.COM/
MAGIC.PNG?
SATURATION=150

HTTP://
IMAGER.INSTAMIKE.COM/
MAGIC.PNG?FLIP=TRUE

HTTP://
IMAGER.INSTAMIKE.COM/
MAGIC.PNG&FLOP=TRUE

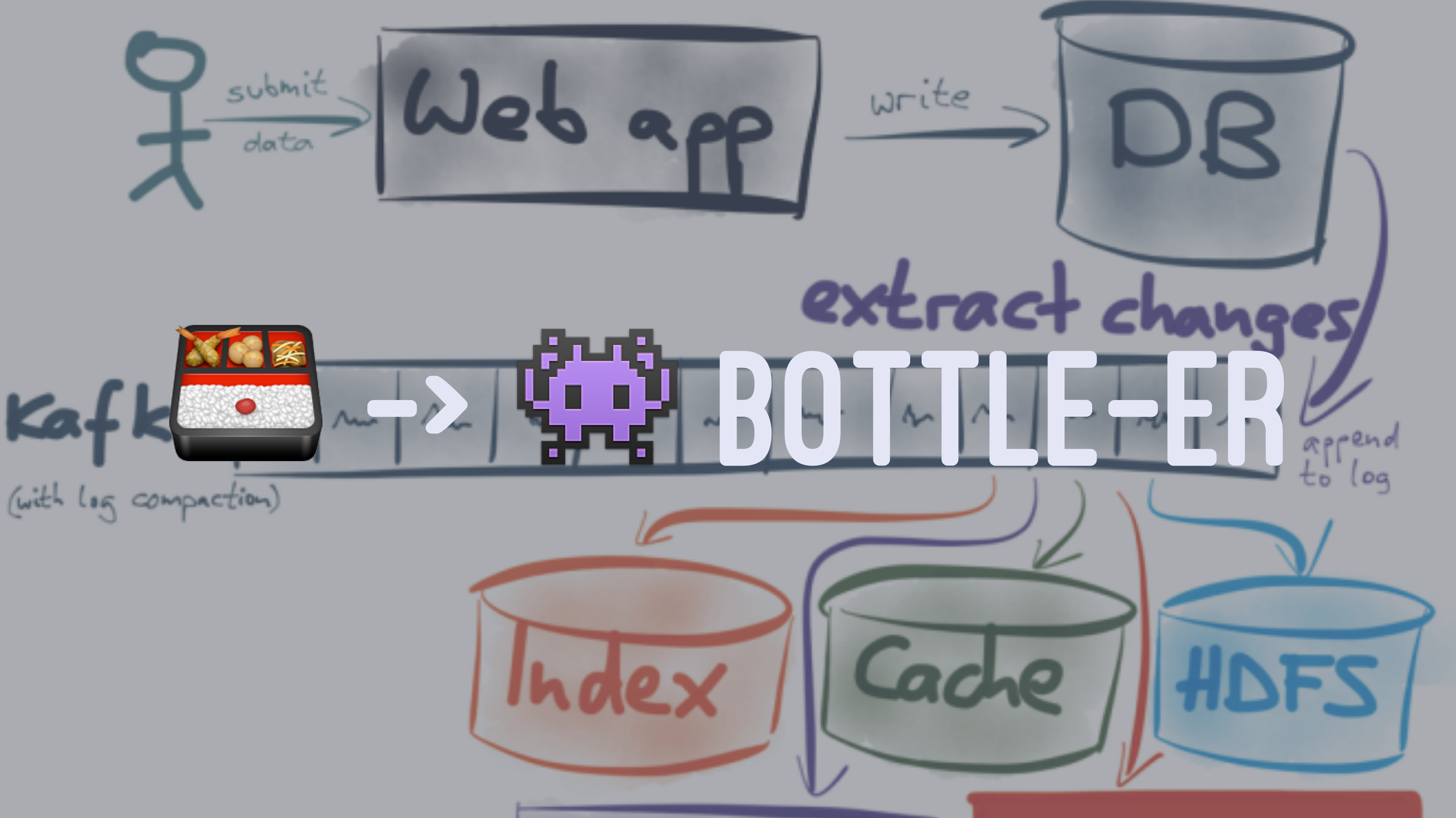
 ->  IMAGE-ER

TRY IT!

- HTTP://BIT.LY/1NSP3VI
- HTTP://MAGICKLY.AFELD.ME/?SRC=HTTP://I.IMGUR.COM/FJ5D8NQ.GIF&FLIP=TRUE
- HTTP://MAGICKLY.AFELD.ME/?SRC=HTTP://I.IMGUR.COM/FJ5D8NQ.GIF&TWO_COLOR=TRUE

ETL CONTEXT -> BOTTLE-ER

- POSTGRES HAS A LOG OF EVENTS
- EVERY TIME YOU DO AN OPERATION IT GETS UPDATED
- IT IS USED TO REPLICATE THE DATABASE
- SO. INSTEAD OF RUNNING A RUBY SCRIPT TO GENERATE REPORTS EVERY NIGHT.


submit
data
Web app
write
DB
extract changes
Kafka
(with log compaction)
append
to log
Index
Cache
HDFS

BOTTLE-ER

 -> 

BOTTLE-ER

INTEGRATION IS REALLY SIMPLE:

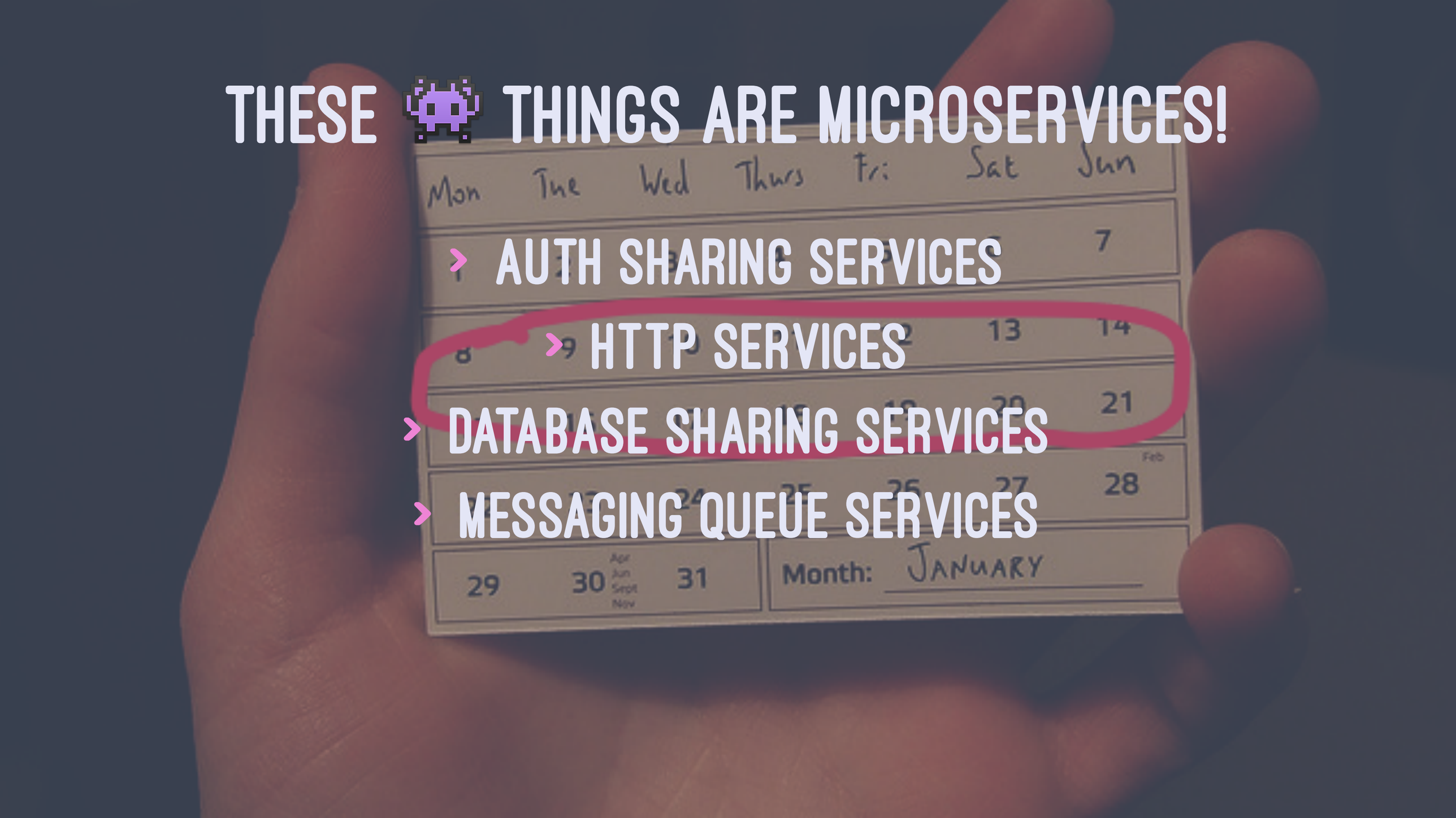
```
ruby
$ docker run -d --name bottledwater --
hostname bottledwater
\ --link postgres:postgres
\ --link kafka:kafka
\ --link schema-registry:schema-registry
confluent/bottledwater:0.1
```

SO WHAT ARE THESE THINGS?


THESE THINGS ARE MICROSERVICES!
HELPER SERVICES (APPS) YOU CAN REWRITE FROM SCRATCH IN 2 WEEKS OR LESS.

THESE 👾 THINGS ARE MICROSERVICES!

- AUTH SHARING SERVICES
- HTTP SERVICES
- DATABASE SHARING SERVICES
- MESSAGING QUEUE SERVICES



MICROSERVICES 👾 CAN:

- BE WRITTEN IN ANY LANGUAGE
- TALK OTHER SERVICES VIA REST, REDIS, RABBITMQ, COOKIES, OR SMOKE SIGNALS
- LIVE INSIDE OR OUTSIDE YOUR APP'S FIREWALL
- CAN SCALE INDIVIDUALLY

IF YOUR APP USES RESQUE. YOU'RE ALREADY USING SERVICES (BUT NOT MICROSERVICES)

TO GET THEM TO BE 👾:

- MAKE THEM SMALLER. THEIR DEPENDENCIES MUST ALSO BE ABLE TO BE REWRITEN IN 2 WEEKS.- ENABLE THEM TO SCALE INDIVIDUALLY (SOME IMPLEMENTATIONS ALREADY ALLOW YOU TO DO THIS).
- STOP THEM LOADING THE WHOLE APP INSIDE THEM

 MICROSERVICES AREN'T PERFECT

THERE ARE MANY GOTCHAS FOR GETTING MICROSERVICES RIGHT:

- DISTRIBUTED SYSTEMS ARE HARD. (PICK SPEED, CONSISTENCY OR AVAILABILITY)
  - DEBUGGING ACROSS MULTIPLE SERVICES CAN BE TRICKY
  - CREATING DEVELOPMENT & PRODUCTION ENVIRONMENTS BECOMES HARDER
- BEWARE OF MICROSERVICE ENVY. USE THEM WHERE THEY ARE

THREE QUICK TIPS BEFORE WRAPPING UP:

- RABBITMQ OR ANY OTHER MESSAGING SERVICE MAKES IT EASIER TO KEEP STATE CONSISTENT. HTTP LOSES MESSAGES VERY EASILY.
- TRY TO MINIMIZE STATE AS MUCH AS YOU CAN. REPLICATION IS VERY COSTLY
- CHECK THE REFERENCES (SPECIALLY SAM NEWMAN'S BOOK)

MUST WIN BUILDS ❤ APPS

I'M GONZALO MALDONADO AND I'M A LEAD ENGINEER AT MUST WIN.
WE BUILD ❤ APPS. (SOMETIMES WITH 👾 MICROSERVICES)

CHECK US OUT AT

MUSTWIN.COM

REFERENCES

- SAM NEWMAN'S BUILDING MICROSERVICES
- HIGH SCALABILITY BLOG
- CONTINUOUS DEPLOYMENT
- RUBY ROGUES
- HTTPS://SPEAKERDECK.COM/ELGONZ/5-WAYS-WE-SCREWED-UP-MICROSERVICES
- ERIC EVAN'S DOMAIN DRIVEN DESIGN