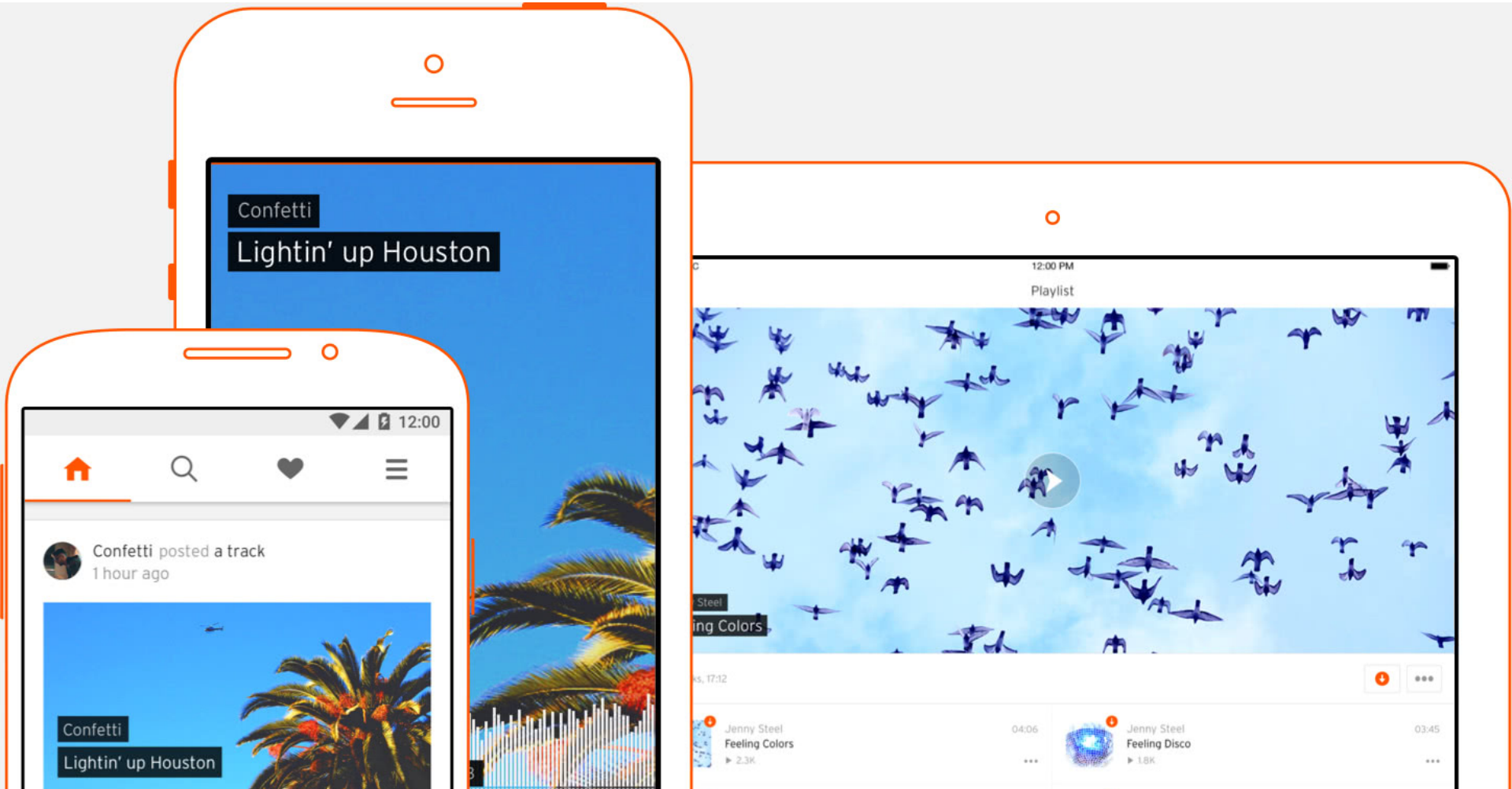
Containerizing your monolith

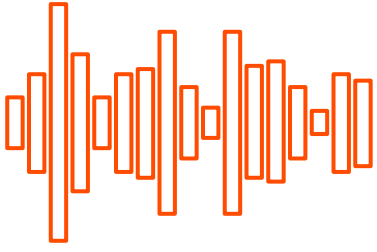
Jano González (@janogonzalez)

Introduction

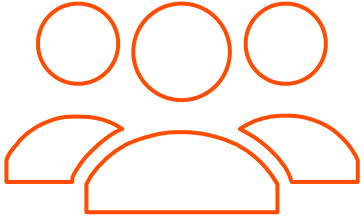
What’s SoundCloud



What’s SoundCloud



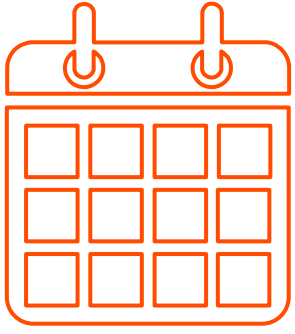
> 200M Tracks



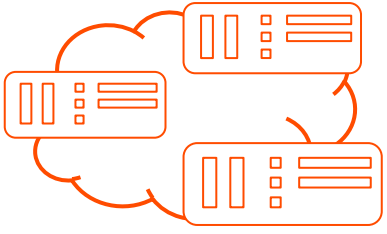
> 20M Creators

Many SoundCloud Rappers

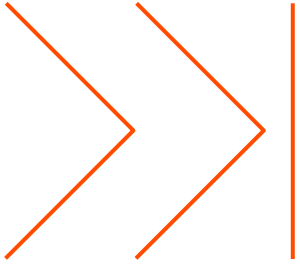
Motivation



2017-2018

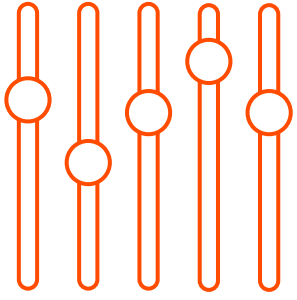


Migration



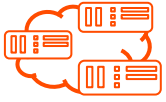
Keep moving

Motivation



What’s the balance?

Contents



Our migration to microservices

The monolith

Containerizing the monolith

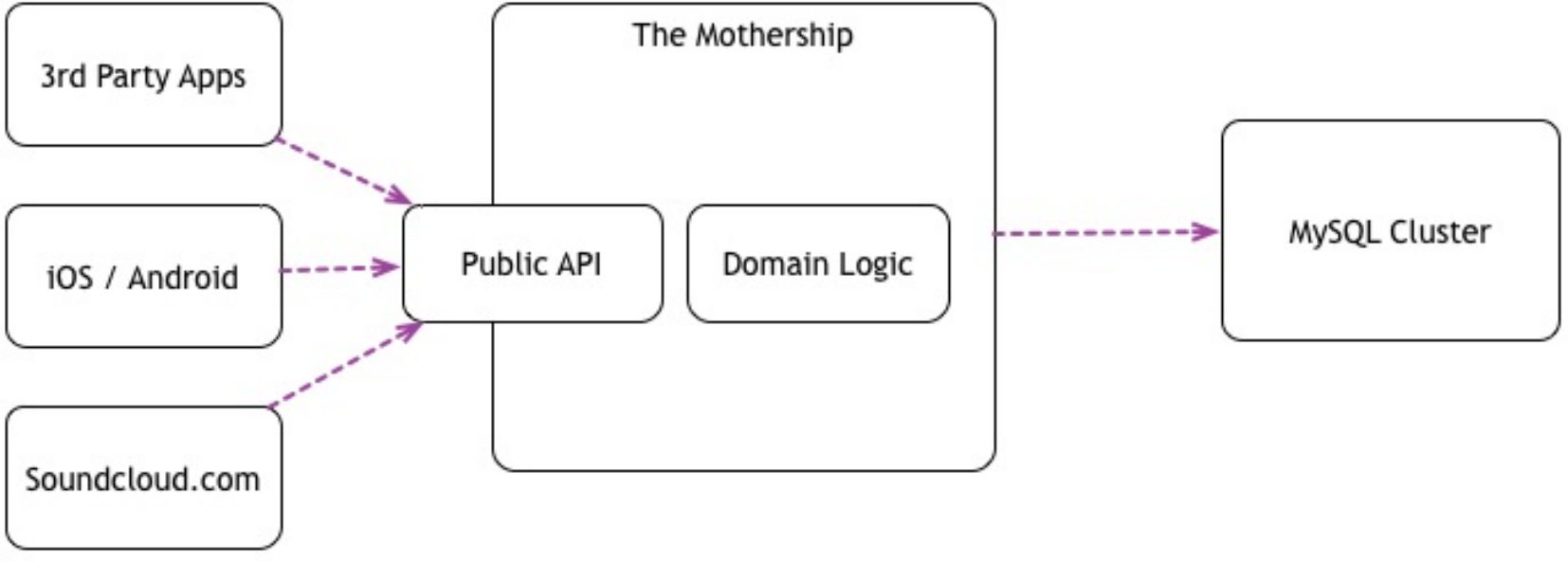
Conclusion

Our migration to microservices

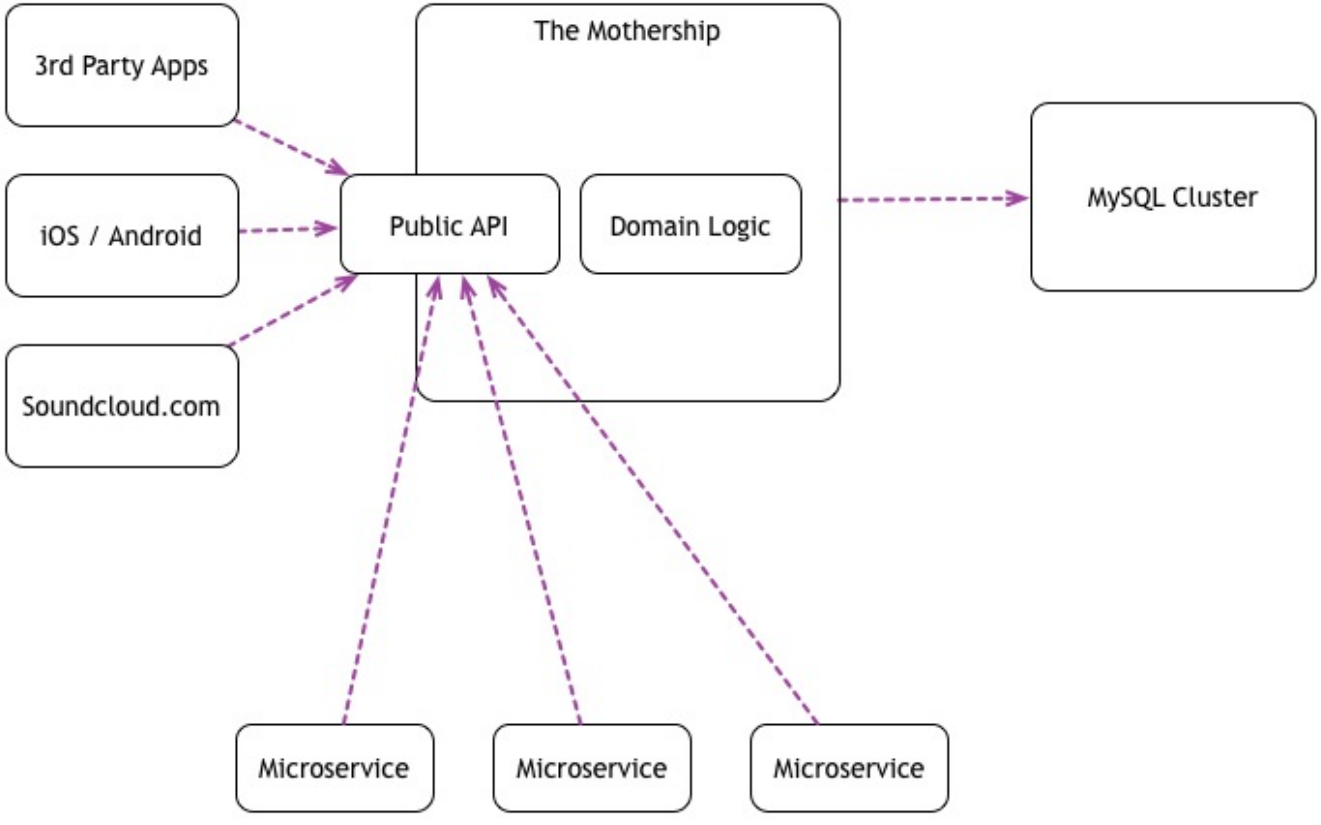
How we started

```
$ rails new soundcloud
```

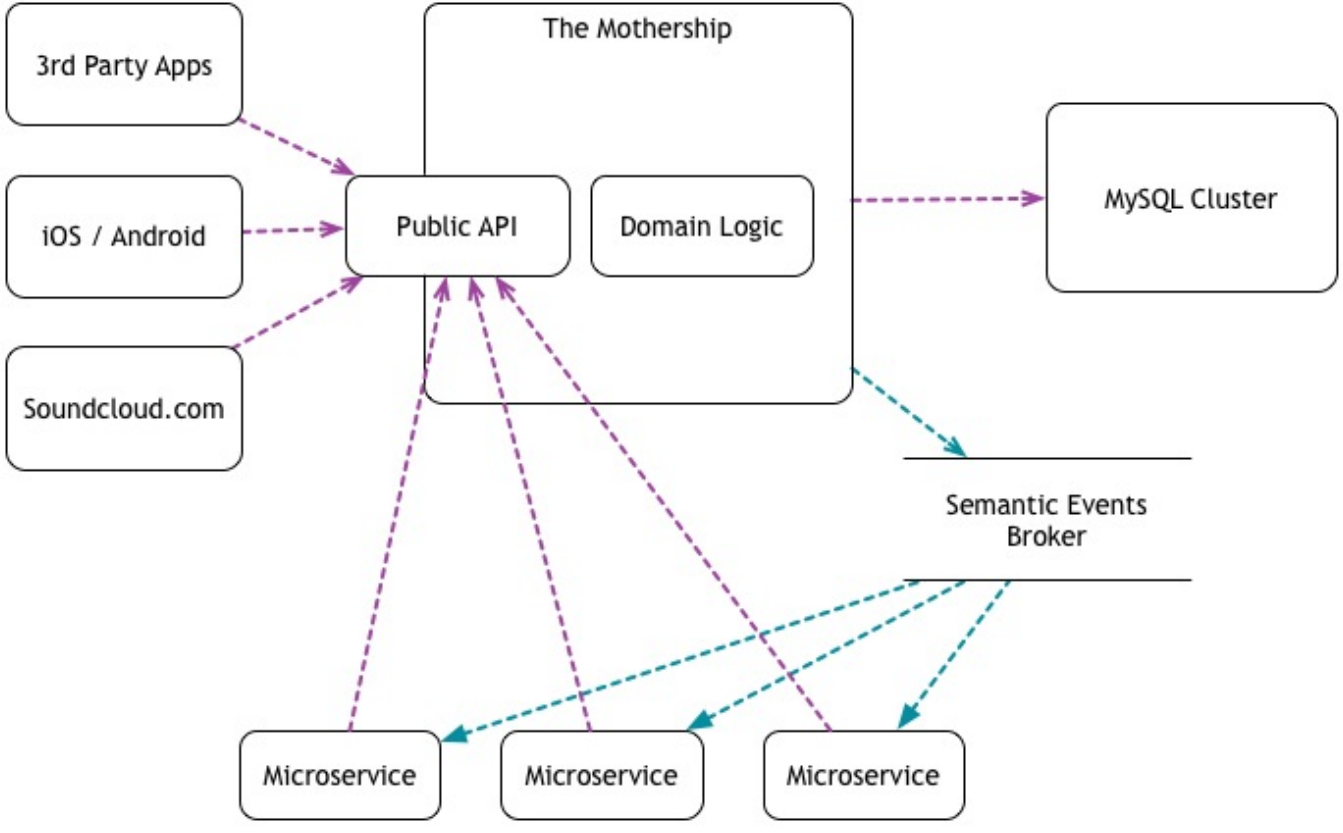
Around 2012


3rd Party Apps
iOS / Android
Soundcloud.com
The Mothership
Public API
Domain Logic
MySQL Cluster

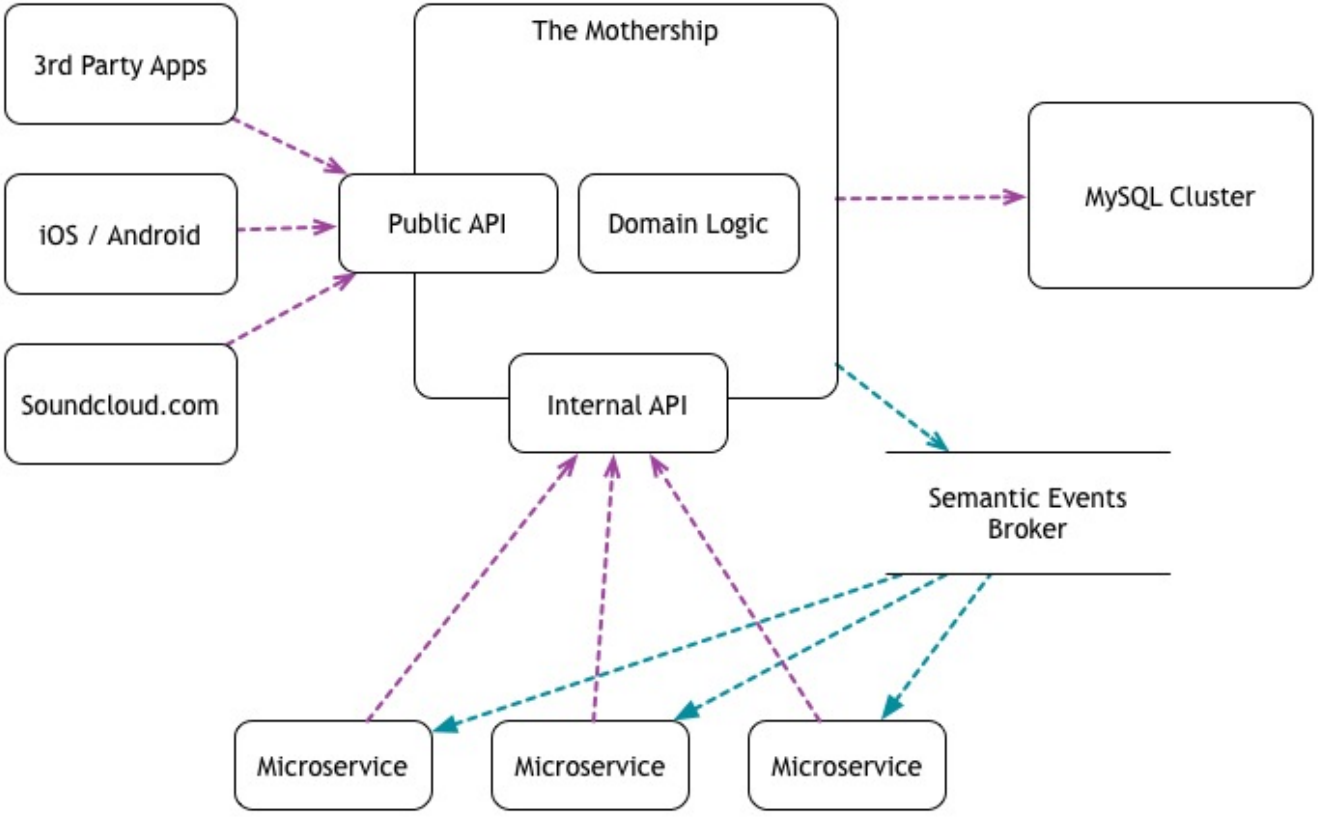
Around 2012


3rd Party Apps
iOS / Android
Soundcloud.com
The Mothership
Public API
Domain Logic
MySQL Cluster
Microservice
Microservice
Microservice

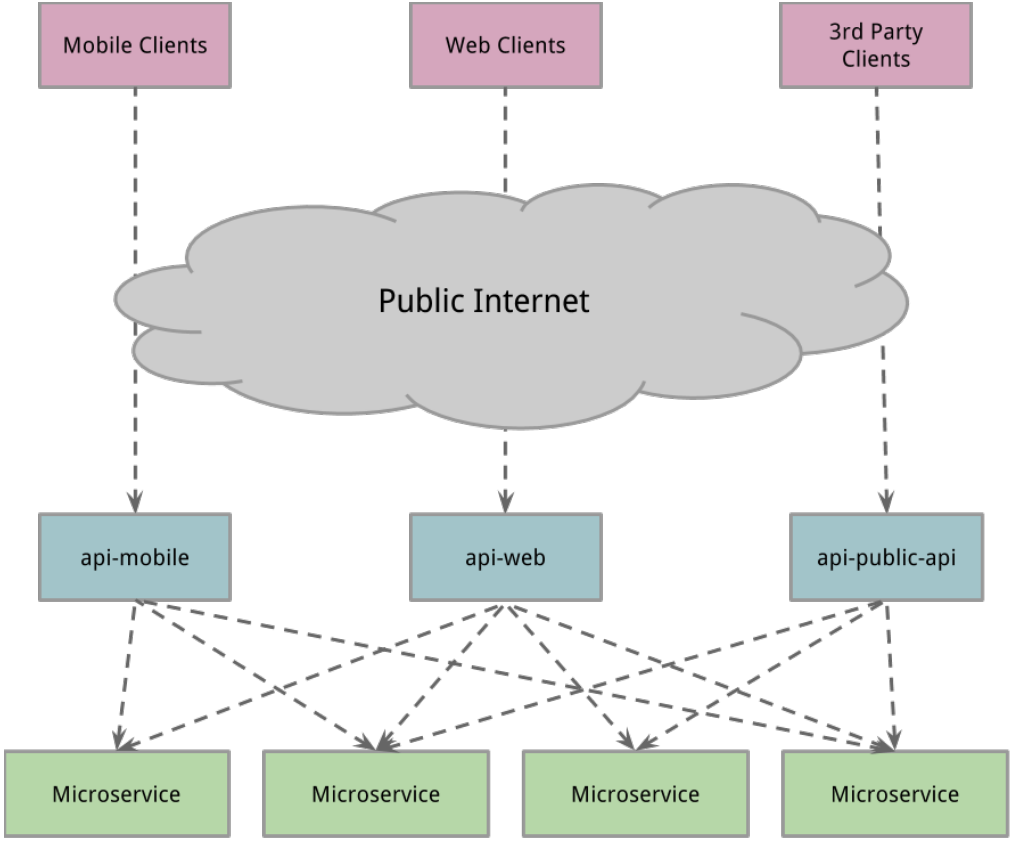
Around 2012


3rd Party Apps
iOS / Android
Soundcloud.com
The Mothership
Public API
Domain Logic
MySQL Cluster
Semantic Events Broker
Microservice
Microservice
Microservice

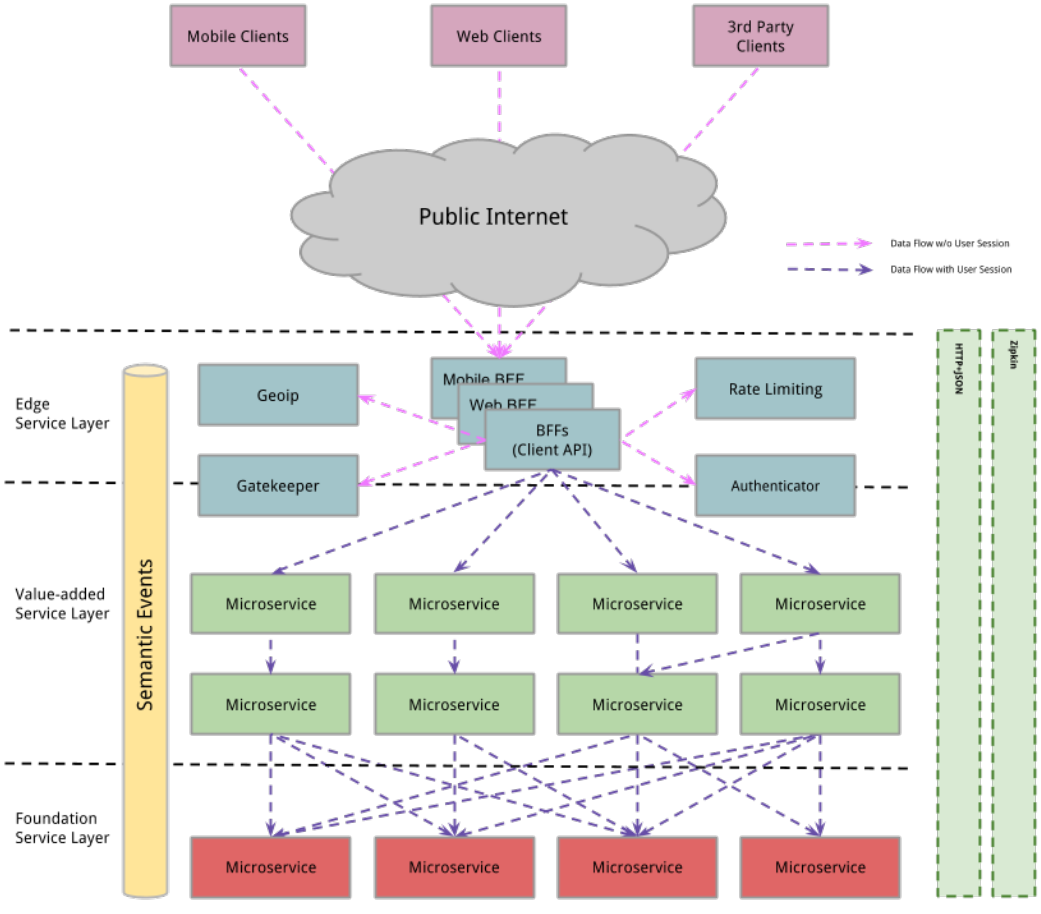
Around 2012


3rd Party Apps
iOS / Android
Soundcloud.com
The Mothership
Public API
Domain Logic
Internal API
MySQL Cluster
Semantic Events
Broker
Microservice
Microservice
Microservice

After


Mobile Clients
Web Clients
3rd Party Clients
Public Internet
api-mobile
api-web
api-public-api
Microservice
Microservice
Microservice
Microservice

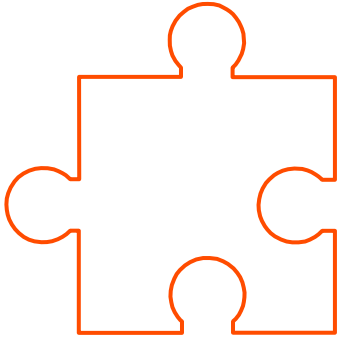
After


Mobile Clients
Web Clients
3rd Party Clients
Public Internet
Data Flow w/o User Session
Data Flow with User Session
Edge Service Layer
Value-added Service Layer
Foundation Service Layer
Semantic Events
Geoip
Mobile BFF
Web BFF
BFFs (Client API)
Rate Limiting
Gatekeeper
Authenticator
Microservice
HTTP+JSON
Zipkin

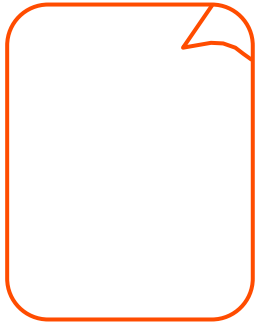
But what about deployment?

Deployment

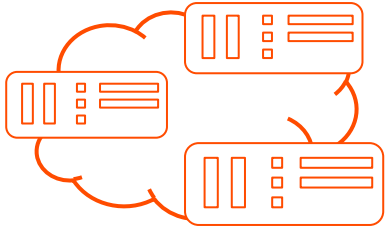
Our abstractions



Component



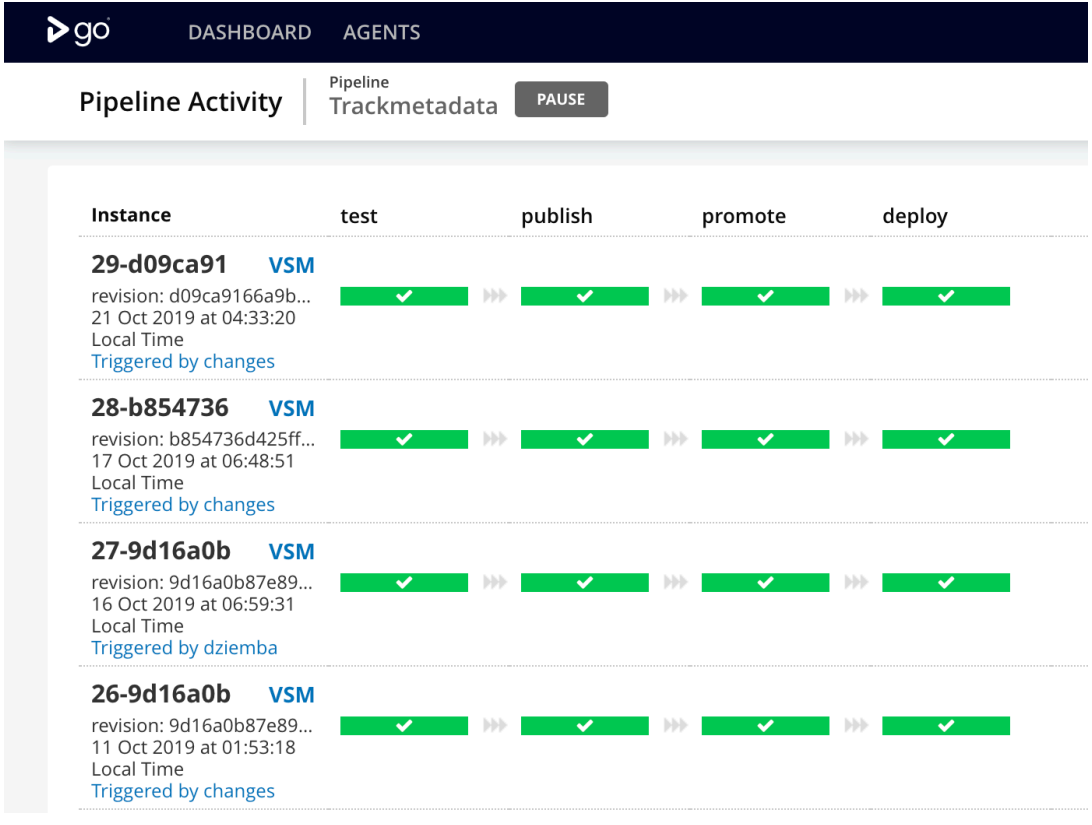
Environment



Zone

Deployment

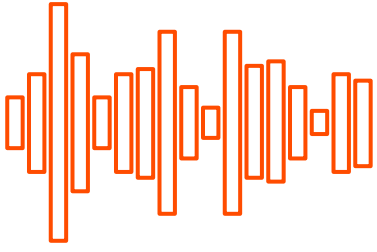
The process



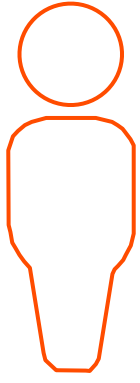
The monolith

The monolith

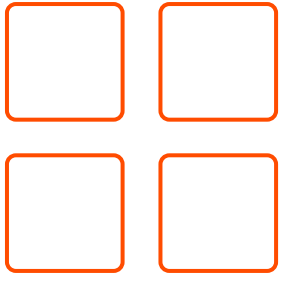
Core entities



Track



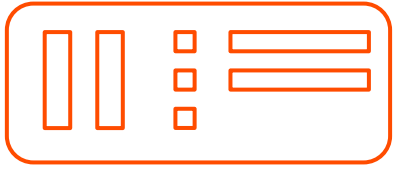
User



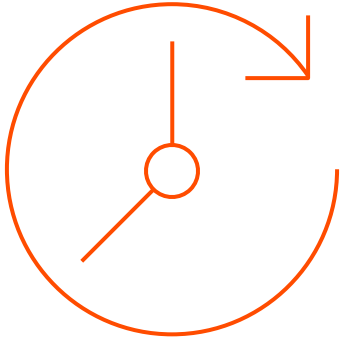
Playlist

The monolith

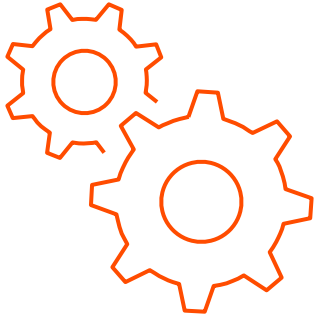
The technology



360 Chef provisioned bare-metal machines



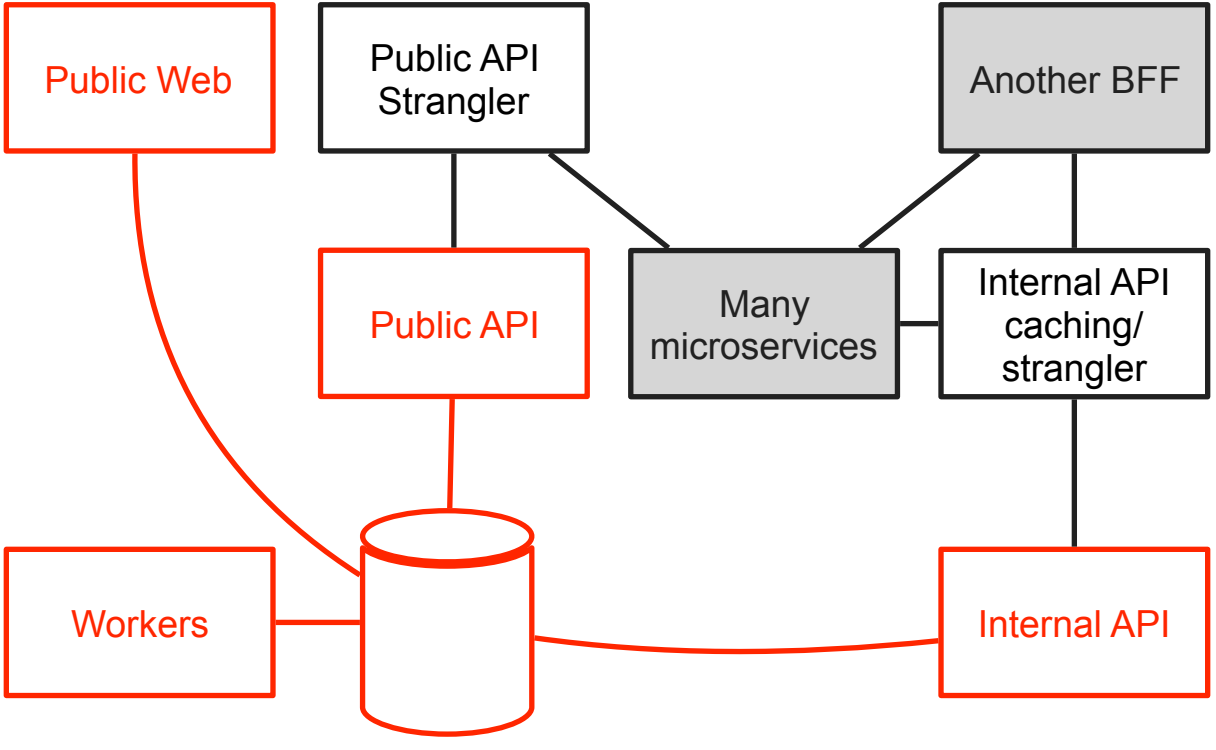
Rails 2.3



Capistrano deployment

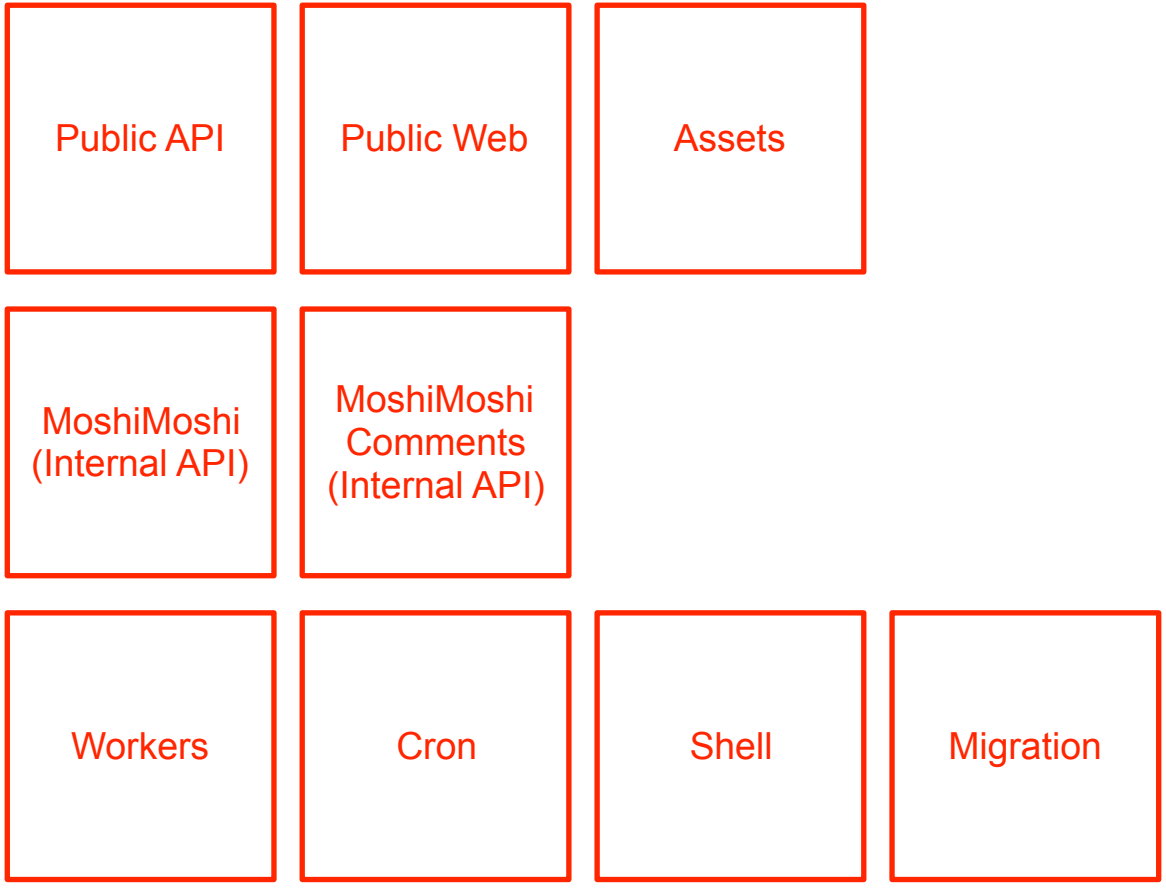
The monolith

The architecture



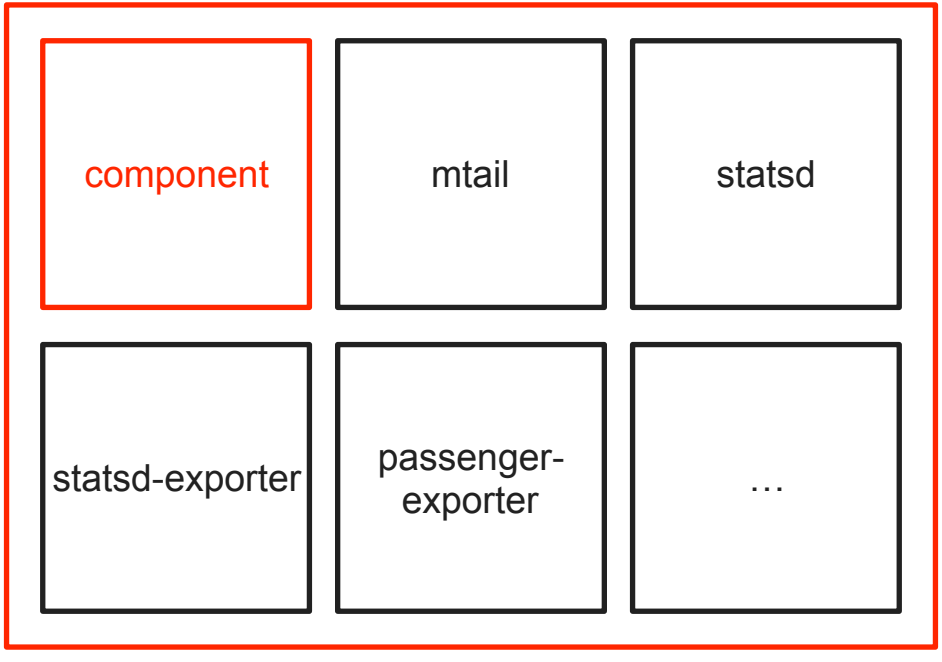
The monolith

The components



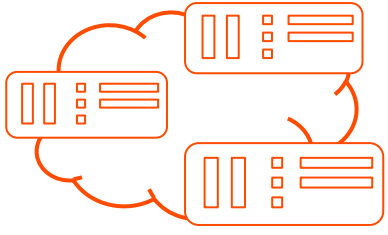
The monolith

The hosts

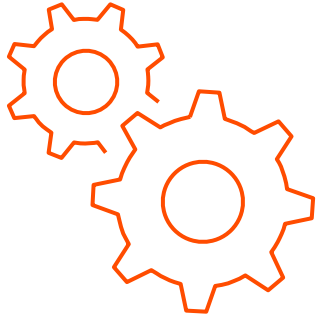


The monolith

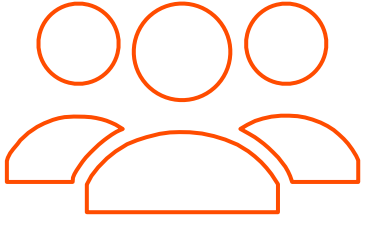
The issues



Utilization



Deployment



Lack of confidence

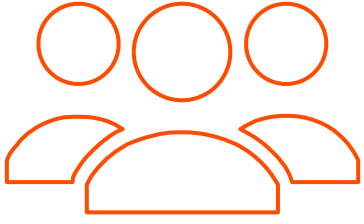
Containerizing the monolith

Throw it into Kubernetes

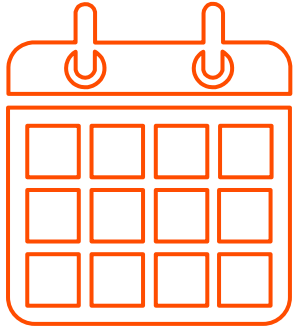
Congrats, your monolith is a microservice now

Thank You!

The project



1.5 Engineers

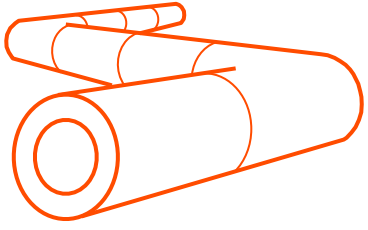


1 year until the last bit was cleaned up

The first milestone

The first milestone

Docker development container

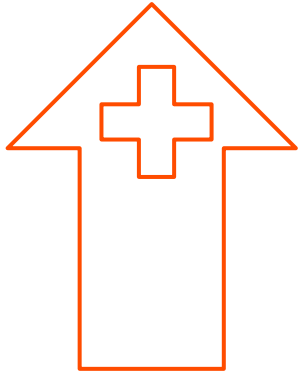


Tests on GoCD

The proof of concept

The proof of concept

Does it even work?

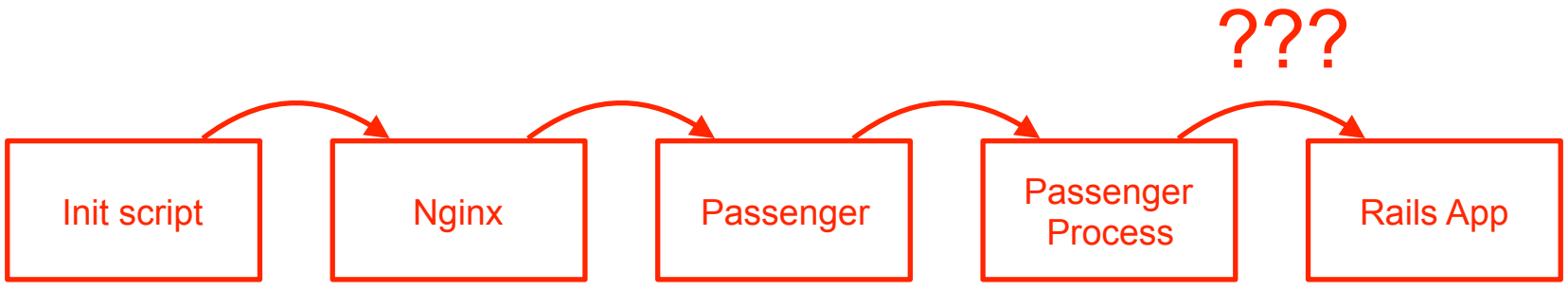


First staging component

PROBLEM!

The proof of concept

Where are my env variables?



SOLUTION

The proof of concept

Env variables with The Perl Hack™

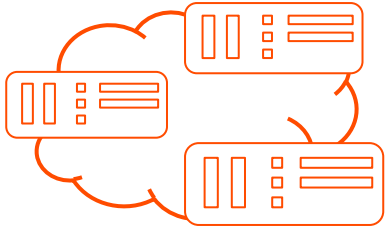
```
5   # Declares env variables
6   <% config.env_variable_names.each do |name| %>
7   env <%= name %>;
8   <% end %>
```

```
61    # Sets a Perl handler for each env variables, to made them available in
62    # passenger without having to write its values on this config file
63  <% config.env_variable_names.each do |name| %>
64    perl_set $<%= name.downcase %> 'sub { return $ENV{"<%= name %>"}; }';
65  <% end %>
```

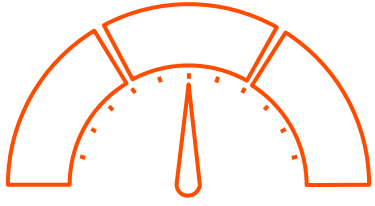
Productionizing

Productionizing

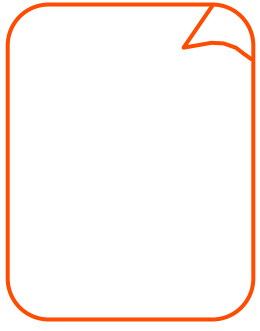
Does it run where it matters?



Deployment



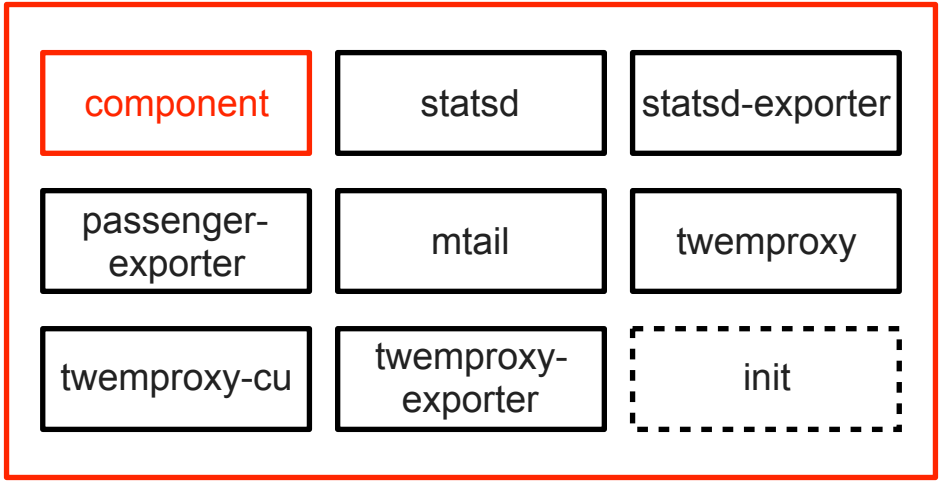
Monitoring



Logs

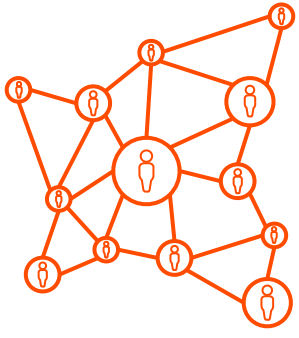
Productionizing

Anatomy of a traffic serving pod

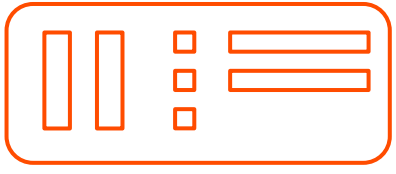


Productionizing

Sizing the pods



Don’t choke service discovery



Be allocatable

Productionizing

Sizing the pods

3

CPU units for main container

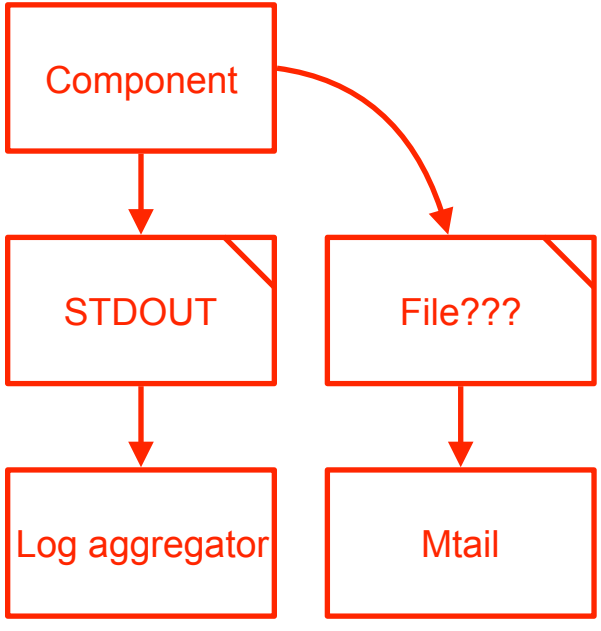
16

Passenger processes

PROBLEM!

Productionizing

Stdout v/s the log metrics exporter



SOLUTION

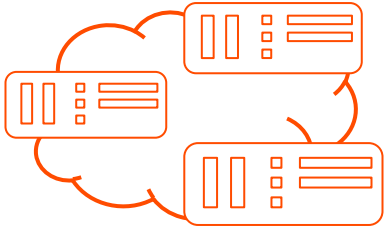
Productionizing

Mtail with The Rotatelogs Hack™

```
11  exec 2>&1
12  exec &> >(rotatelogs -e -L "/tmp/stdout.lnk" -n 2 "/tmp/stdout.log" "50M")
13
14  exec /usr/sbin/nginx \
15    -c /srv/mothership/app/config/nginx/nginx-moshimoshi.conf \
16    -g "daemon off;"
```

Productionizing Public API

Productionizing Public API

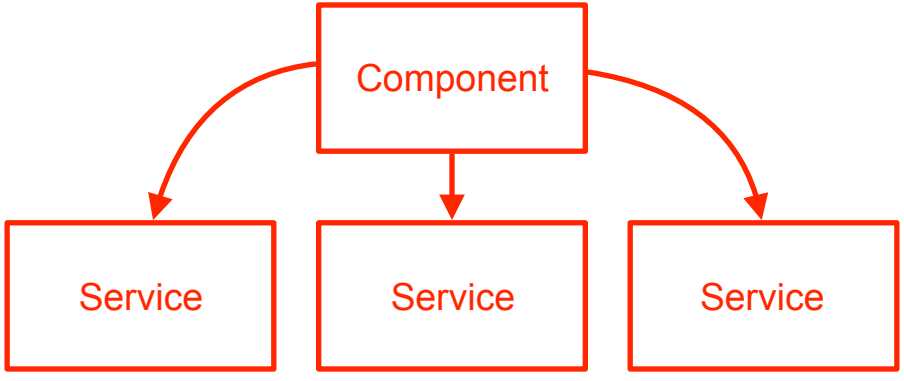


Orchestration

PROBLEM!

Productionizing Public API

DNS latency and our excessive usage



SOLUTION

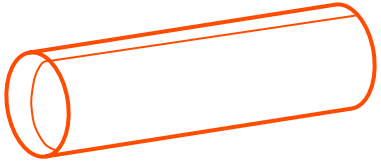
Productionizing Public API

CoreDNS and the DNS Hack™

```
6   class Resolv
7     class DNS
8       class Config
9         # Internal optimization, we consider those domains absolute
10        FORCED_ABSOLUTE_DOMAINS = /\.(com|net|io)\z/
11
12        def generate_candidates(name)
13          candidates = nil
14          if FORCED_ABSOLUTE_DOMAINS =~ name
15            name = "#{name}."
16          end
17          name = Name.create(name)
```

Productionizing Internal API

Productionizing Internal API

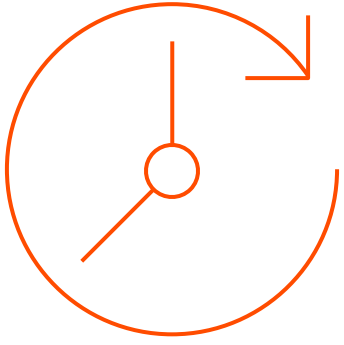


Highest throughput

PROBLEM!

Productionizing Internal API

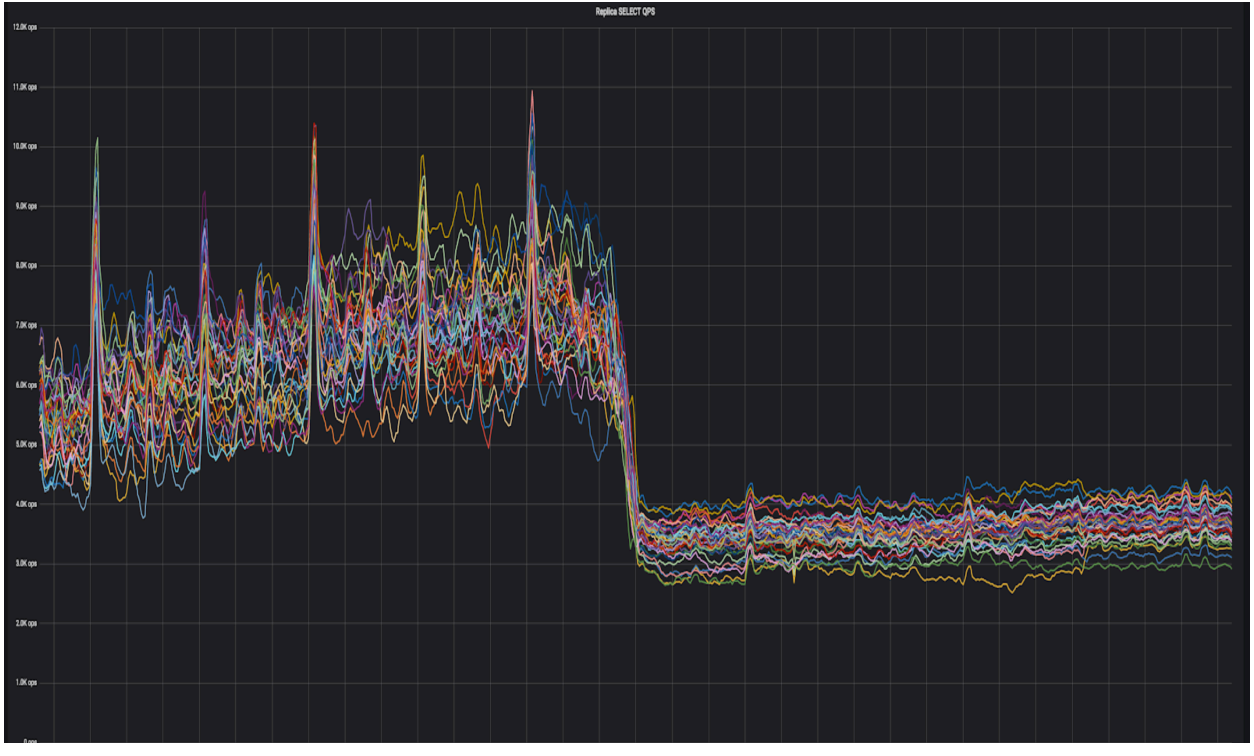
Latency was too high



SOLUTION

Productionizing Internal API

Optimize GC and make cheaper SQL queries



PROBLEM!

Productionizing Internal API

Errors spikes during deployment

SOLUTION

Productionizing Internal API

The preStop Trick™

```
lifecycle:
  preStop:
    exec:
      command: ["/bin/sh", "-c", "sleep 15 && /usr/sbin/nginx -c /srv/mothership/app/config/nginx/nginx-{{ component }}.conf -s quit"]
```

PROBLEM!

Productionizing Internal API

Errors spikes during deployment (still???)

SOLUTION

Productionizing Internal API

The Pre Start Trick™

```
readinessProbe:
  httpGet:
    path: /-/health
    port: 8200
  initialDelaySeconds: 60
```

```
passenger_pre_start http://<%= config.hostname %>:8200/-/health;
```

The rest

The rest

- Workers
- Cron jobs
- Shell / Migration hosts
- Cleanup!

Finishing

Remove capistrano deploy #4259

Merged **janogonzalez** merged 2 commits into `master` from `remove-capistrano-deploy` on 12 Sep 2018

Conversation 0 | Commits 2 | Checks 0 | Files changed 25

janogonzalez commented on 7 Sep 2018 • edited

Removes the Capistrano deployment. Now the `config/database.yml` and `config/migration_database.yml` are versioned.

This can be merged when the migration node is ready for the new migration procedure.

2

Current status

Current status

Number of pods

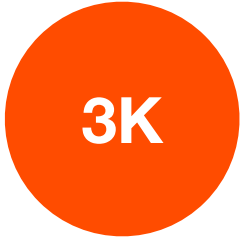
On-prem

Cloud

Current status

Traffic

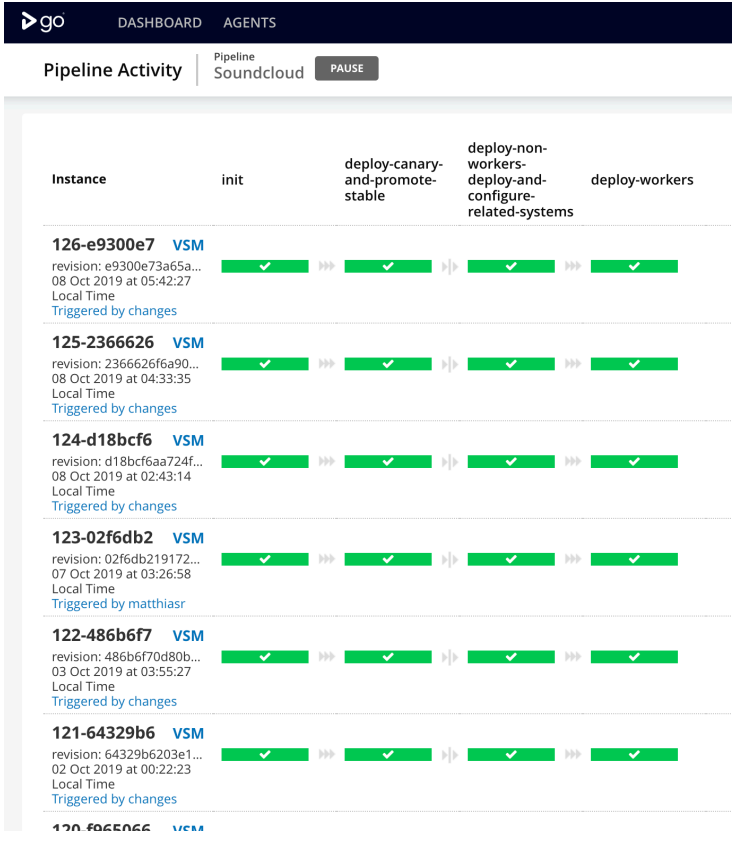
On-prem RPS



Cloud RPS

Current status

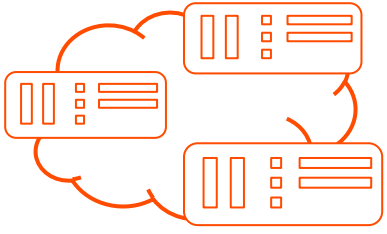
Many deploys



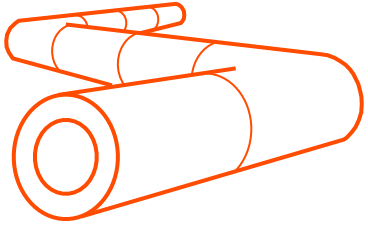
Conclusions

Conclusions

What we solved



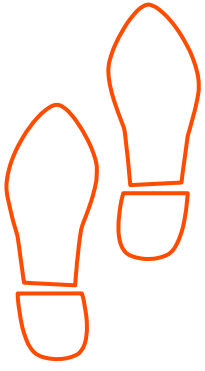
One Infrastructure



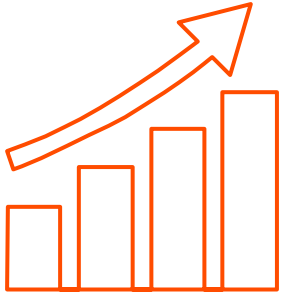
One Delivery Process

Conclusions

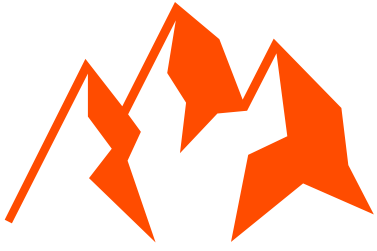
How we did it



Step by step



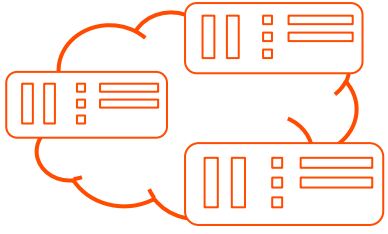
Controlled rollouts



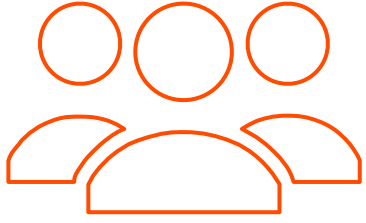
Managing expectations

Conclusions

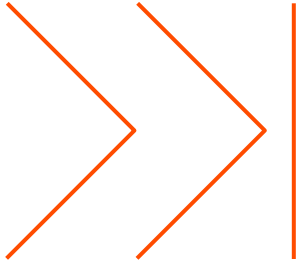
Benefits



Improved utilization



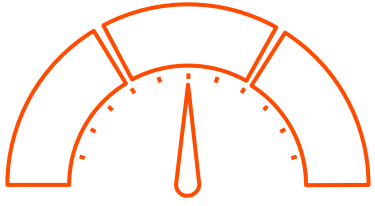
Increased confidence



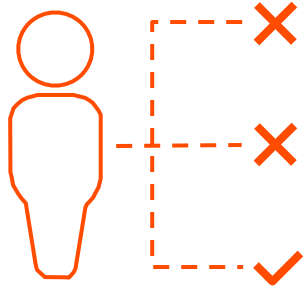
Enabling new initiatives

Conclusions

Should you do it?



Assess current progress



Evaluate costs and benefits

Thank You!

@janogonzalez

https://soundcloud.com/janogonzalez