

RED HAT
SUMMIT

How to handle the complexity of migrate to Microservices from Monolithic 10 years of code

Reasons to being or not involved in migrating to Microservices

Alberto Salazar,
CTO Advance Latam

4th May 2017

Who am I?

Alberto Salazar



www.advlatam.com

asalazar@advlatam.com



www.ecuadorjug.org



www.cloudbanco.com



@betoSalazar
@EcuadorJUG

- Developer -> Architect -> Speaker
- Founder and CTO: AdvanceLatam & Cloudbanco
- Involved on a C level for the last 10 years
- Working 17 years with monolithic & continue writing code
- Founder and Leader EcuadorJUG
- JCP Associate Member & JSR's early adopter

Agenda

- The use case - The evolution of 10 years of code
- The motivation to move from Monolith to Microservices
- The fact - 10 years of code
- The path - Tips, tricks, pros & cons of Microservices
- Summary

The evolution of 10 years of code

The use case



JEE 5 (2005)
JBOSS 4.2.2.GA
JSF, EJB3, EIP & others
(Facade, DAO, DTO
Services Delegate, etc).



SOA (2007)
Industry approach
ESB
BPM
BAM
(-) Not succeed
external influence



OSGi (2011)
Any App Server

SpringDM
Apache Camel
Apache ActiveMQ
Html 5 Css3 Mobile



OSGi Blueprint
(2013)
Apache Karaf
AMQ
(Split the Front-end
from the back-end)



Microservices
(2014)
-> Split the monolith
Elasticsearch
Logstash
Hazelcast

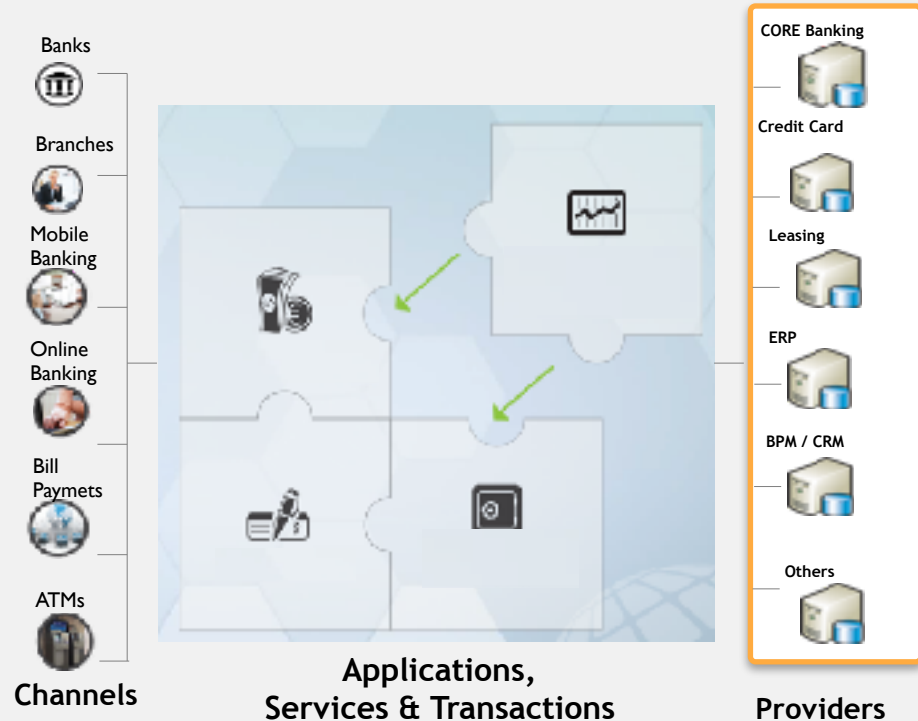


Microservices
(2016 - 2017)
+ Vert.x | Spring boot
+ Apache Cassandra
+ NodeJS, ReactJS & React Native

The business

You have to understand the sponsor

- First -> Multichannel
- Lately -> Omnichannel
- Now -> Digital



The motivation to migrate from Monolith to Microservices

Why microservices ?

The buzz



Microservices is the architectural approach that everybody talks about and everybody wants it, but be careful

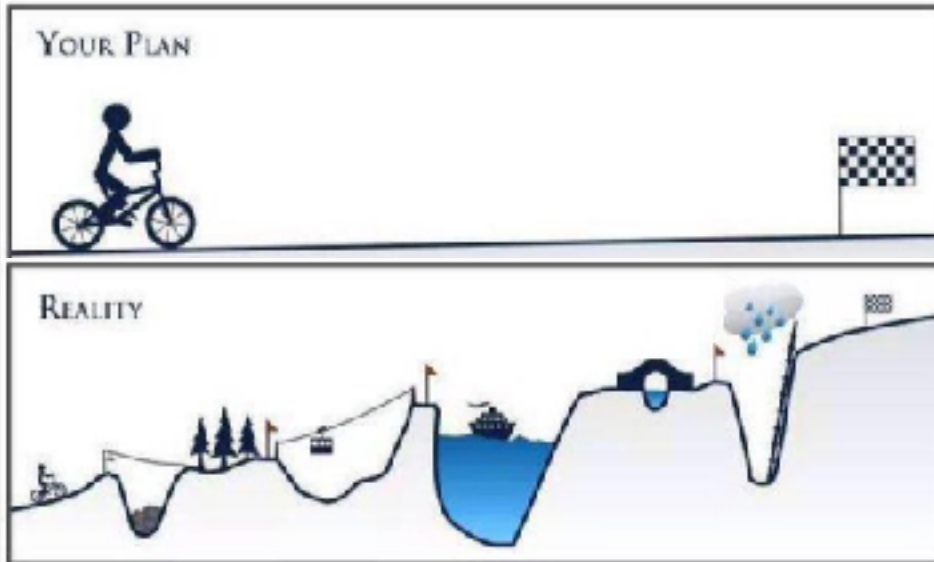
The Goal



The goal that organizations need is to increasing velocity & agility;
Get into production as soon as possible;
Deploy new features as soon as possible.

The reasons of this talk

The plan of move forward (Microservices ?)



Every body are talking about the result (microservices architectural style)
but nobody are showing the **path**

The fact - 10 years of code

The Monolithic

The Monolith

Characteristics



- Attachment to language, platform & OS
- Single logical executable, deploy everything at once or nothing at all
- Failure of part == failure of whole
- Take months even years getting into production
- Centralized authority slows the delivery process (DBA, OPS, QA)
- Coordinated releases are hard, because brings many changes together from different teams

1 Year of code

Always the goal was to keep the modularization



12 ejb-jars, 8 wars, 15 developers, JEE 5, build by 2 Teams (The framework & The Business)

5 Years of code

Modularization -> Osgi bundles



Around 50 osgi-modules, 3 wars, 1 ear, 40 developers, JEE + SpringDM, organized in around 5 Teams (The framework, Mobile, The Business, 3 Customization team's)

10 Years of code

Modular WTF



More than 100 osgi-bundles + more than 50 jars modules, 1 ear + 5 wars, Modularize JS resources as Jars, >100 Developers, JEE + Spring + Apache Karaf, Camel, AMQ, 10 to 15 Teams (framework, Mobile, Business App, Customer's team)

10 Years of code



Application Metrics

Lines of Code

2 169 233
0 (NotMyCode)

Types

60 225
278 Projects
4 961 Packages
539 523 Methods
198 640 Fields
0 Source Files

Migrate 10 years of code, It will be Easy ?

The challenge



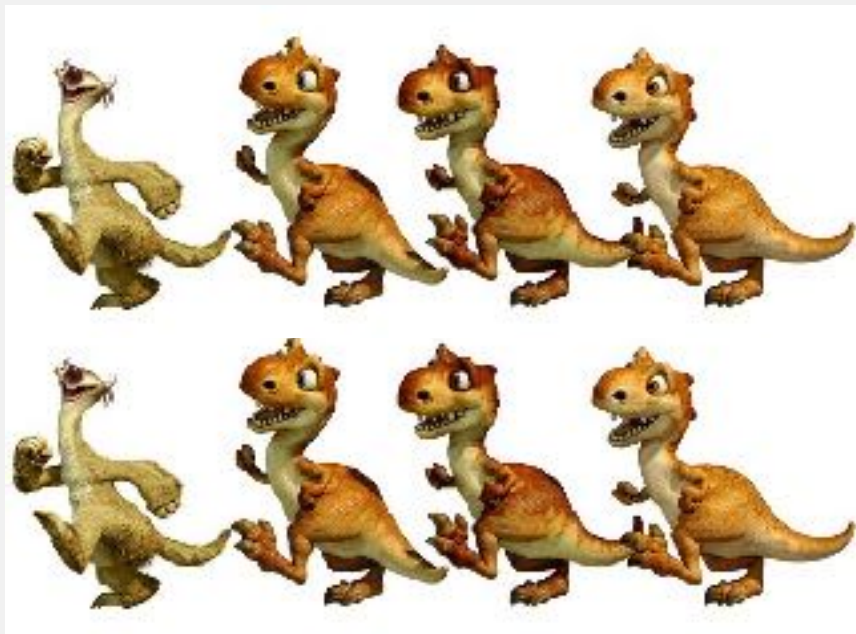
- > 2MM lines of **highly** coupled code
- Build one microservice is, easy but what about a complete microservice architecture based on 2MM lines of code
- Time to delivery features are between: 6 to 12 months & QA overhead \$\$
- Several customers using the system on production environment

The path - Tips, tricks, pros , cons

Microservices

Microservices

Characteristics

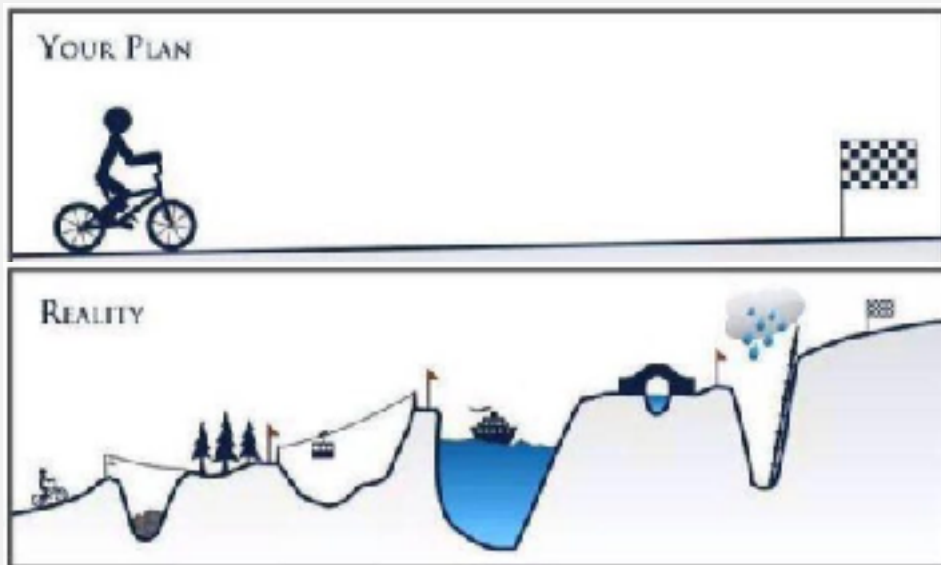


<https://martinfowler.com/articles/microservices.html>
<https://martinfowler.com/bliki/MicroservicePrerequisites.html>

- Independently deployable & executable
- Based on services
- High cohesion, low coupling
- Failure is isolated
- Model driven design
- Effective & efficient scaling
- Polyglot “Plus”

The path

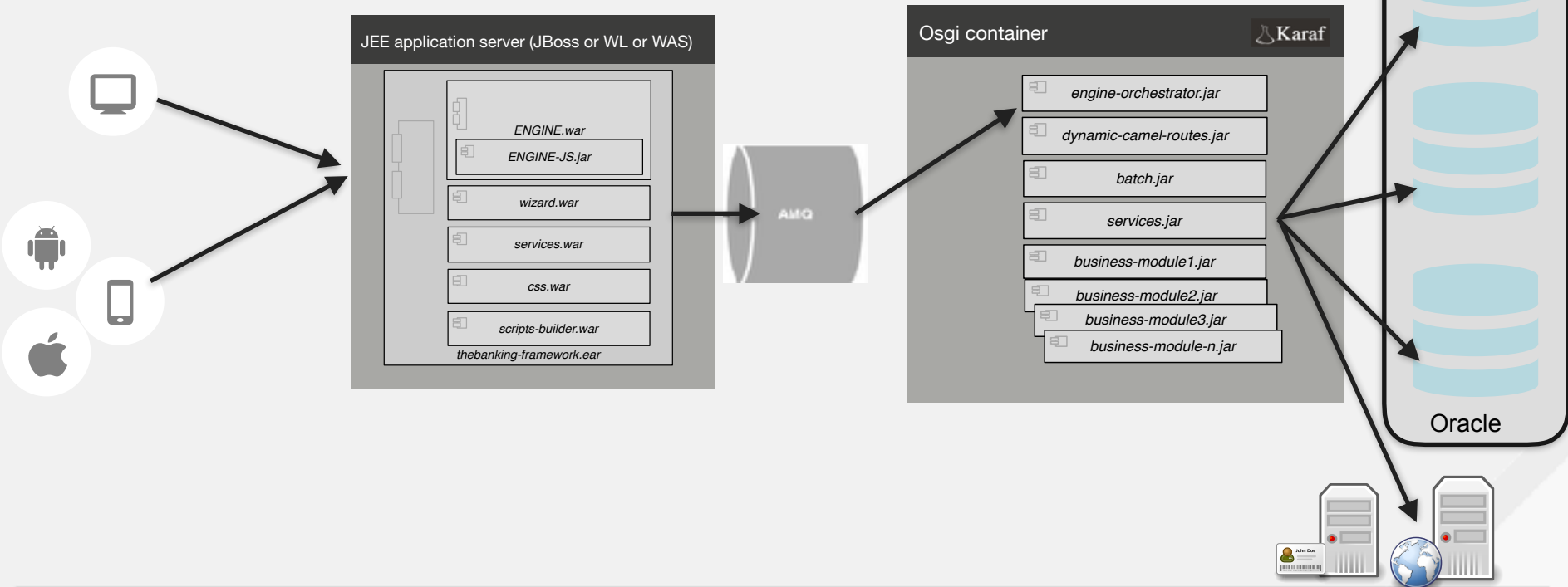
Microservices



- Split the frontend from the backend
- Split the backend & keep the centralized data
- Rest endpoints & the API Gateway
- Split the data
- Don't forget the frontend & the agility

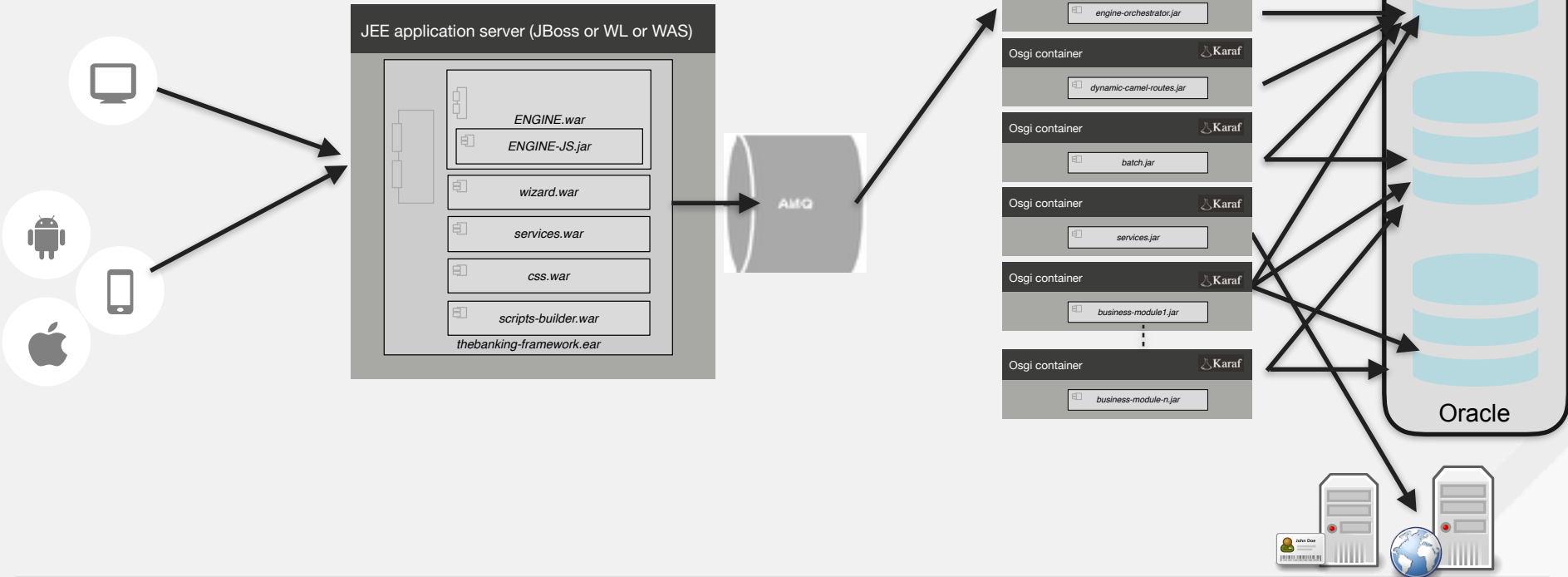
The path

Split the frontend from the backend (1/5)



The path

Split the backend (2/5)



The path

Split the backend (2/5)



OSGi container Karaf
engine-orchestrator.jar

OSGi container Karaf
dynamic-camel-routes.jar

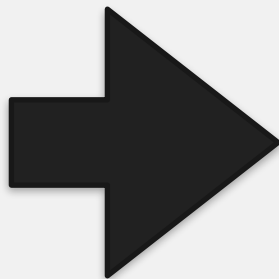
OSGi container Karaf
batch.jar

OSGi container Karaf
services.jar

OSGi container Karaf
business-module-1.jar

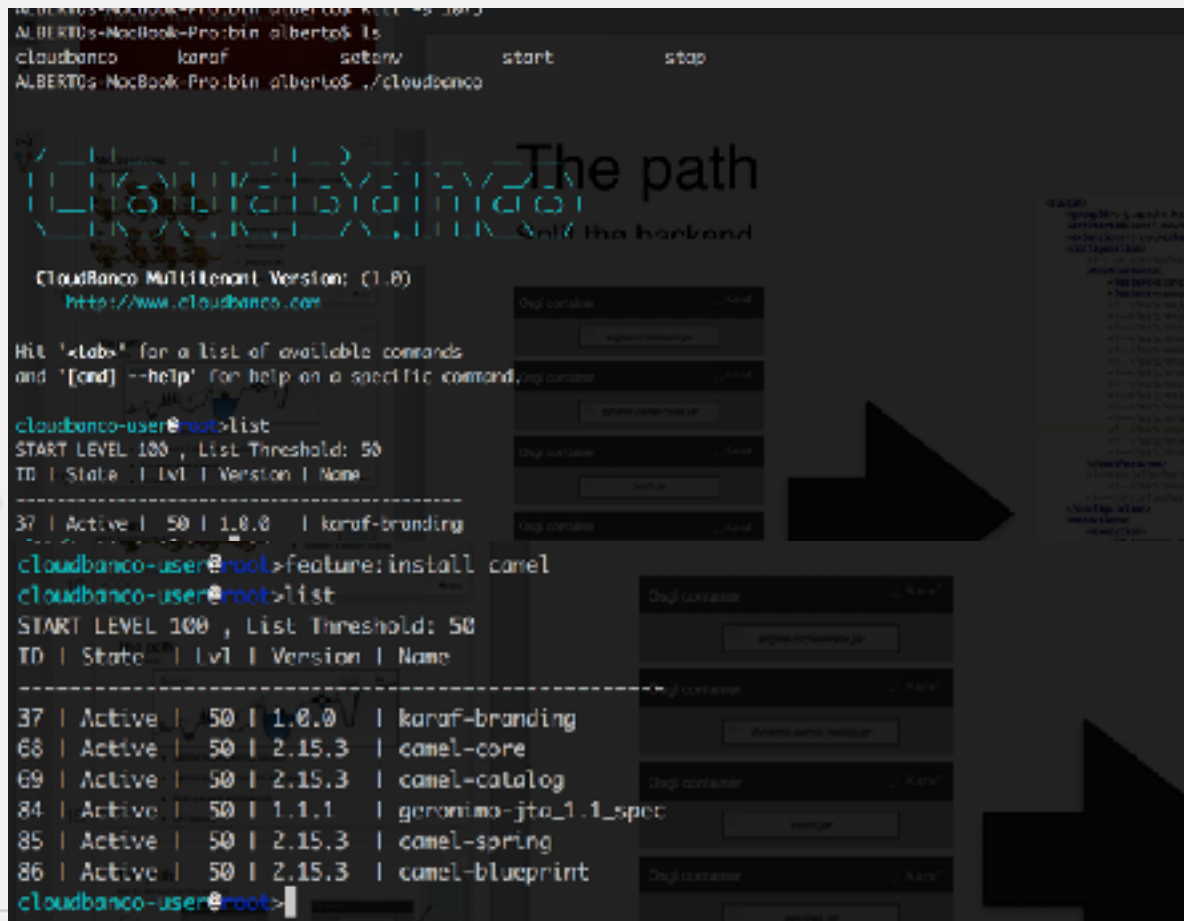
OSGi container Karaf
business-module-n.jar

Apache Karaf
with a custom
distribution



install just
what you need

https://karaf.apache.org/manual/latest/#_custom_distributions



```
ALBERTO's-MacBook-Pro:bin alberto$ ./cloudbanco
cloudbanco karaf setenv start stop
ALBERTO's-MacBook-Pro:bin alberto$ ./cloudbanco

The path
Split the backend

CloudBanco Multitenant Version: (1.0)
http://www.cloudbanco.com

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.

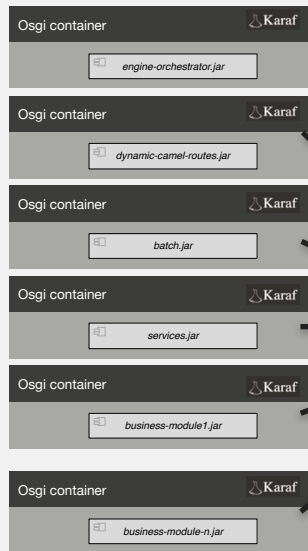
cloudbanco-user@root>list
START LEVEL 100 , List Threshold: 50
TD | State | Lvl | Version | Name
-----
37 | Active | 50 | 1.0.0 | karaf-branding

cloudbanco-user@root>feature:install camel
cloudbanco-user@root>list
START LEVEL 100 , List Threshold: 50
TD | State | Lvl | Version | Name
-----
37 | Active | 50 | 1.0.0 | karaf-branding
68 | Active | 50 | 2.15.3 | camel-core
69 | Active | 50 | 2.15.3 | camel-catalog
84 | Active | 50 | 1.1.1 | geronimo-jta_1.1_spec
85 | Active | 50 | 2.15.3 | camel-spring
86 | Active | 50 | 2.15.3 | camel-blueprint

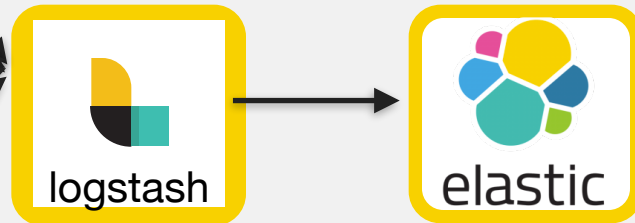
cloudbanco-user@root>
```

The path

Split the backend (2/5) - Logging, trace & Monitoring



- 1) Use Mapped Diagnostic Context (MDC)
- 2) Introduce a correlationId
- 3) Collect the logs
- 4) Search by rest API or use Kibana

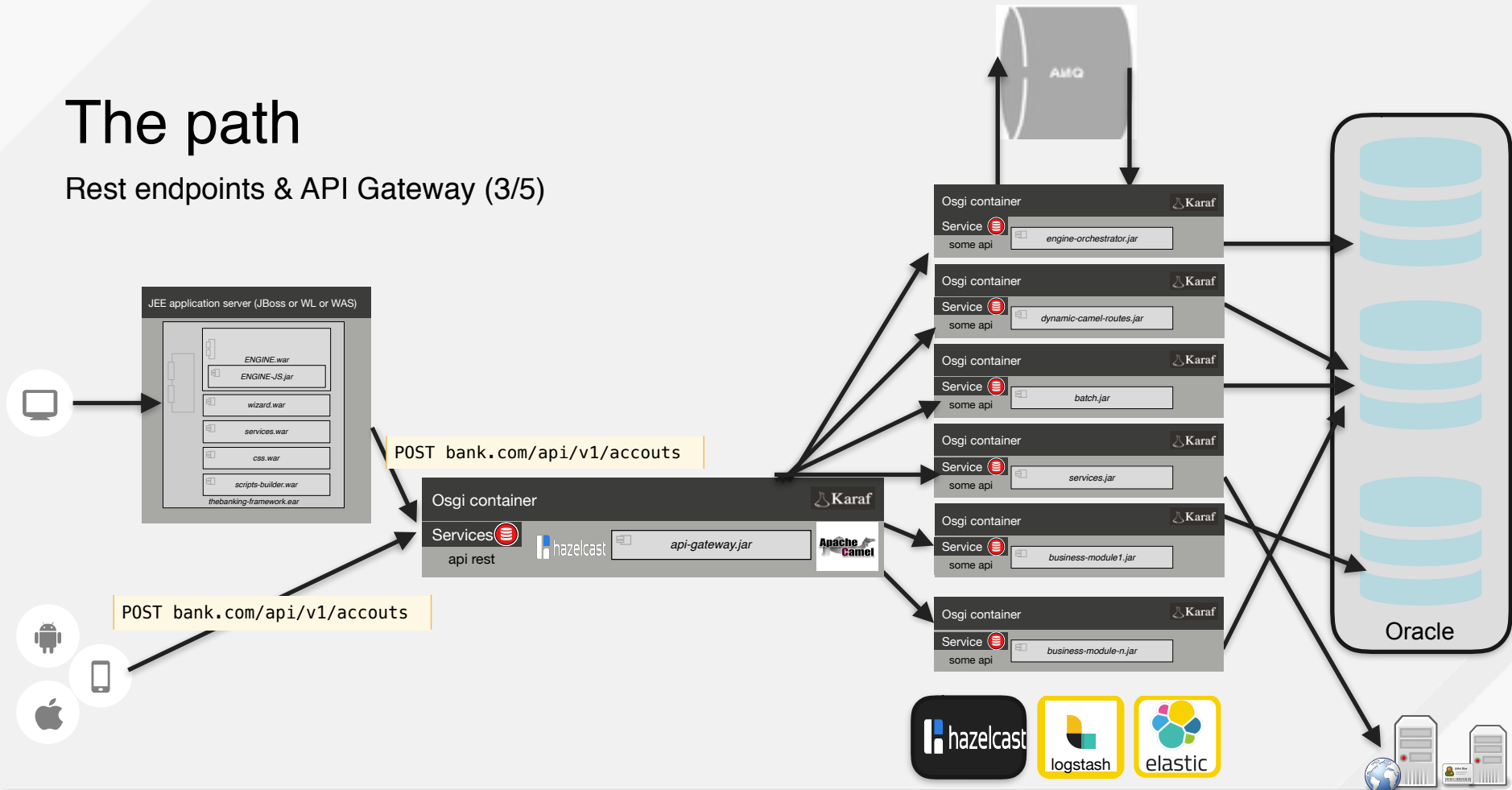


<http://camel.apache.org/mdc-logging.html>
<http://www.baeldung.com/mdc-in-log4j-2-logback>
<https://www.elastic.co/products/elasticsearch>
<https://www.elastic.co/products/logstash>
<https://www.elastic.co/products/kibana>

```
logstash {
  type => "java"
  codec {
    json_lines {
      meta_key => "@version"
    }
  }
  filters {
    gsub {
      pattern => ["%"]
      replacement => "\\%"
    }
  }
  outputs {
    elasticsearch {
      hosts => ["localhost:9200"]
      type => "java"
      codec {
        json_lines {
          meta_key => "@version"
        }
      }
    }
  }
}
```

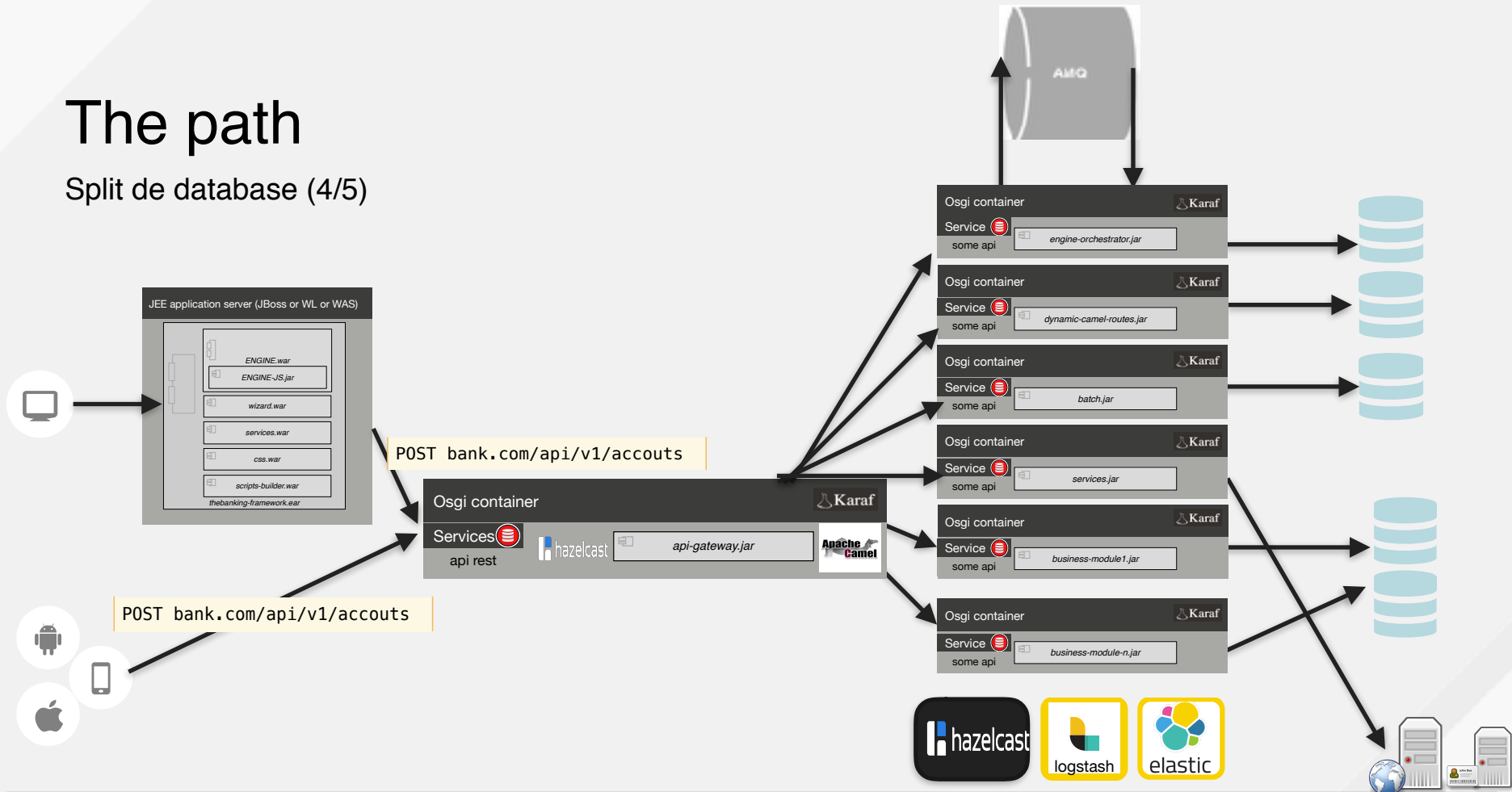
The path

Rest endpoints & API Gateway (3/5)



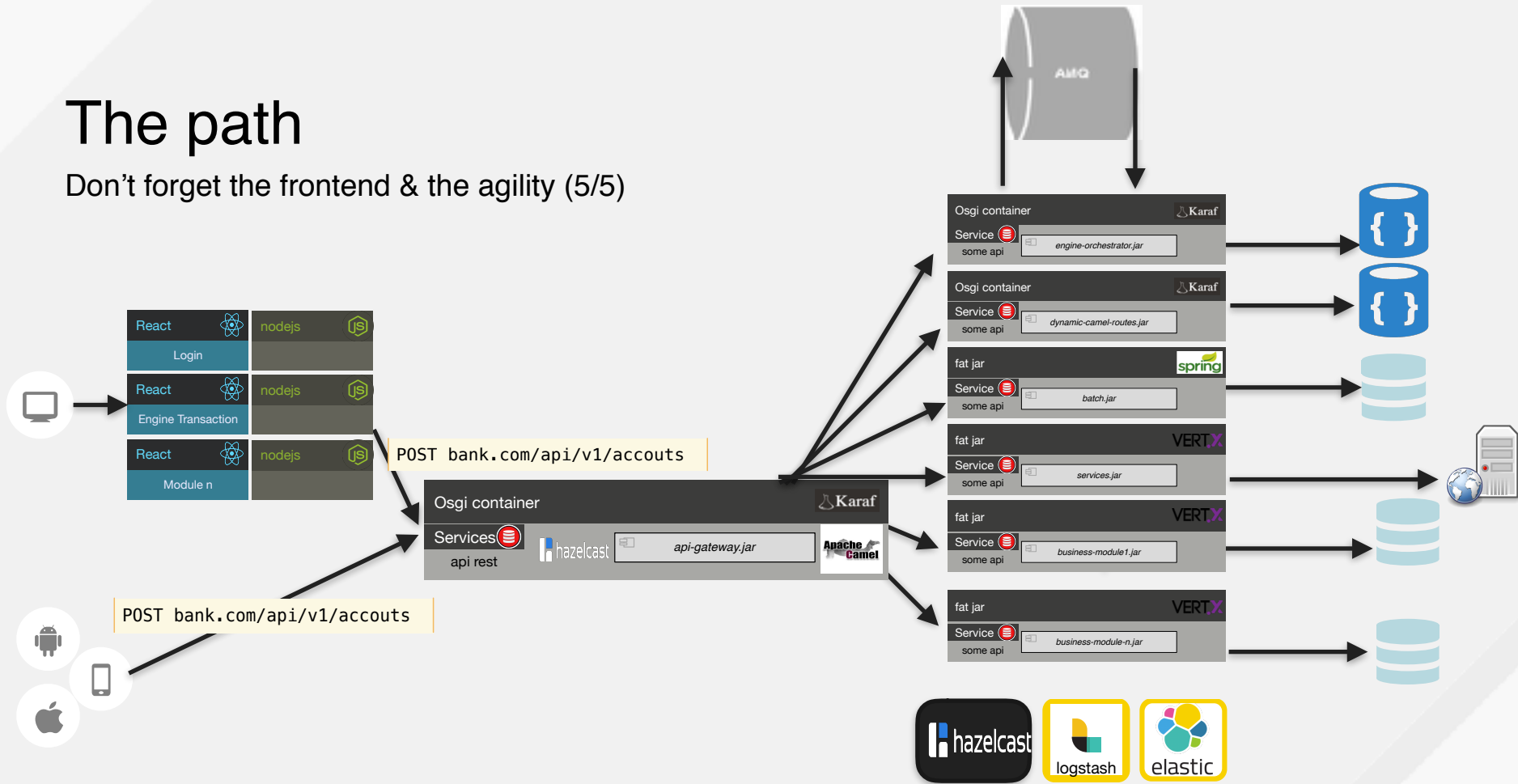
The path

Split de database (4/5)



The path

Don't forget the frontend & the agility (5/5)



Summary & Code blueprints

Summary

Microservices Architecture Losses



- Transactions
- Single data repository
- Better architects needed
- Greater complexity - it bears repeating
- the system and services have to deal with network communications, failures, rebalances, splits.
- Monolithic apps are far easier to develop and debug (when viewing the platform as a whole)

Summary

Recommendations

- ✓ Design your application modular (either monolith, OSGi or microservices)
- ✓ Continuously refactor your modules or micro services to achieve optimal boundaries
- ✓ Define your remote and async APIs carefully, design remote calls for failure
- ✓ Monolithic apps only look simple from the outside, but you just open the box
- ✓ A lot of help using Event Driven Architecture (decoupled, scalable, Competing Consumers Patter)
- ✓ Care about logs, monitoring and always use a CORRELATIONID and MDC

Summary

Recommendations

- ✓ Each team will be able to explore and test new technology
- ✓ Automate the deployment and delivery process -> CI & CD
- ✓ Split the database
- ✓ Use lightweight frameworks or java containers (Karaf, Vert.x, Spring boot)
- ✓ Microservices are not everyone, be careful
- ✓ Design for failover, Service load balancing and automatic scaling, data Separation, Integrity, Performance

code blueprints

<https://github.com/lasalazarr/fastdev>



THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHatNews



youtube.com/user/RedHatVideos

RED HAT
SUMMIT

**LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.**