Challenges to internationalize the largest digital bank of the world
Who are we?
Nubank is the leading fintech in Latin America, born to eliminate complexity and empower customers to take control of their money.
Credit Card 2014
Account 2017
Lending 2019
GROWING RAPIDLY IN A COMPLEX DOMAIN

- **Customers**: 20M
- **Engineers**: 500
- **Kafka Messages/day**: 1B
- **Microservices**: 350
- **Deploys/day**: 10s
- **Shards**: 15
- **Teams**: 50
- **HTTP Requests/day**: 1B
- **2014** to **2020**

Graph showing growth in customers, with an exponential increase from 2014 to 2020.
Internationalize!

Late 2018
Mexico Beta
Credit Card

September, 2019
Why talk about i18n?
Agenda

1. INFRASTRUCTURE JOURNEY
2. SYSTEM JOURNEY
3. TEAM BUILDING JOURNEY
4. CONCLUSION
Infrastructure Journey
BACK IN 2014

Ruby Scripts and Cloud Formation

Microservices, continuous delivery, immutable infrastructure since the beginning
BACK IN 2015

Ruby Scripts and Cloud Formation with Docker
BACK IN 2016

Infrastructure sharding

Whole infrastructure, per customer, not just the db
Copies of infrastructure with a global shard to do
customer agnostic operations or routing
SCALABILITY UNITS + GLOBAL ROUTING

- SERVICE 1
- SERVICE 2
- SERVICE 3
- shard S0
- SERVICE 4
- SERVICE 5
- SERVICE 6
- shard S1
- SERVICE 1
- SERVICE 2
- shard S2
- SERVICE 1
- SERVICE 2

purchase → global routing → deposit
BACK IN 2017

Ops automation

Clojure deployment scripts
Infrastructure EDN definition repository
NuCLI
Real time Monitoring
```json
{:datastores
  {:s3
    {:buckets
      [{:config-key "s3_bucket"
         :bucket-name "nu-feed"
         :env-attached? false}]}
    :redis
    {:clusters
     [{:config-key "ratelimit_redis" :cluster-name "rtlimit"}]}
    :datomic
    {:databases
     [{:config-key "datomic_uri" :transactor "feed-1" :name "feed"]}]
    :deprecated
    {:datomic_memcached true
     :kafka true
     :type :jetty
     :port 8081
     :scaling_policies
     [{:cpu_high_alarm
       :cpuvery_high_alarm
       :cpu_ultra_high_alarm
       :cpu_low_alarm
       :cpu_ultra_low_alarm}]
    :environments
    [:prod #nu/prototypes-for [:prod :sharded]
     :staging #nu/prototypes-for [:staging :sharded]
     :test #nu/prototypes-for [:test :sharded]}
```
nu services

Nu services is out of date, run 'nu update' to pull the latest version.

Available commands:
- admin
- api-version
- bump
- channel
- consumers
- container-version
- cssh
- curl
- cycle
- debug
- dump
- edit-secret-config
- git-version
- ip
- launch-config
- list
- manifest
- ops-health
- owner
- pin-curl
- ping
- promoted
- protection
- right-size
- routes
- scale
- shards
- ssh
- versions

nu services cycle

Nu services is out of date, run 'nu update' to pull the latest version.

Cycle instances for a given service. This script will double the desired capacity on the
autoscaling group then set a scheduled update action decreasing the desired capacity back to
normal in 15 minutes.

Usage:
  cycle [options] <service>
  cycle [options] <prototype> <service>

Options:
- --suffix SUFFIX Needed to disambiguate when 2 or more stacks are running
- --region REGION The aws region where the command will be executed. If not given, it will try to guess using the informed env.
- --env ENV The environment where this command will run. (default: prod)
- --reason REASON The reason why this command is being executed so people can be aware of it.
- --help Show help options.
BACK IN 2018

Kubernetes
Better Infrastructure Definition repository
New shards happening too often
BACK IN 2019

Stability

16 shards, better monitoring, Prometheus, Grafana, SLIs, SLOs, OpsGenie
Internationalizing Infrastructure
DECISION

One AWS account per country vs Single AWS Account
Kubernetes is not magic

It’s not only deploy in another AWS account
Legacy Ruby Scripts vs Clojure Scripts

Resources that we only had to create once:
Network setup, memcached clusters, Zookeeper, etc
CHALLENGES

Everything automated but lots of BR specific scripts

Pointing directly to specific buckets, specific resources, etc

Add br/ prefix in everything
Dealing with multiple AWS credentials

NuCLI
Security keys and certificates

IAM, scopes and users
Global identity system
CHALLENGES

ETL

Aggregating data from multiple countries
Hack to launch Mexico while building a better structure that will solve for any country

Manual AWS account setup took several weeks to run

Huge empirical work to figure out what was not properly i18n’ed
System Journey
BACK IN 2014

Started as a front-end to a Credit Card processor
Back in 2015-2016

Full Credit Card processor in-house

Most of it reverse engineering the other system

Learned the domain on the go
Multiple products: Credit card and Checking account

Redesign the app to support multiple products

Allow customers without Credit Card
Rapid growth

Most of the development was to support it

All the corner cases happened

Tech debt only accruing interest

Optimization of individual flows
Internationalizing System
Front-end of 3rd party again, or adapt the current system?
CHALLENGES

Find the differences of the Credit card product in MX

Credit card looks almost the same in all countries.
We didn’t know what was different
Obvious differences like: Timezone, currency, translations, id docs
If locale = Brazil -> x
If locale = Mexico -> y
...

SOLUTION
Not so obvious differences like: Revolving credit calculations, 3rd party integrations, payment integrations
Microservices: replacing some services by a different one for MX, following the same contract
CHALLENGES

Products need to be able to evolve independently
Platform services are generic
Product services are country specific

Good even for new products in Brazil
Team Journey
One Single team
BACK IN 2015-2016

Squads (Spotify Model)

Theme focused
BACK IN 2017

Packs (Project focused teams)
BACK IN 2018

Tribes

Business Units
Groups of Squads
BACK IN 2019

Distributed teams

Teams in Berlin, São Paulo, CDMX
90% of the team in São Paulo
Internationalizing Team
Do all in BR or have a MX/i18n team?
Create a new i18n team that would do all work and no other team would be affected
Onboarding was tough: not enough documentation, hidden BR product rules
I18n teams will mostly build product services, and sometimes help growing the platforms.
Conclusion
Infrastructure Journey

1. Hack
2. Launch
3. Refactor to best architecture
System Journey

1. Hack
2. Launch
3. Refactor to best architecture
Team Journey

1. Hack
2. Launch
3. Refactor to best team organization
CONCLUSION

If there is no refactor, 3rd, 4th countries should be way harder

If the system was not built with i18n in mind, chances are that it won't be easy to do
Interested in joining our revolution?

https://nu.careers
Offices in São Paulo (Brazil), CDMX (Mexico), Buenos Aires (Argentina), Berlin (Germany) and Washington DC (USA)
Remote is an option too.