Rapid Hypothesis Testing Cycle in Large-Scale Web Applications by Rails
Speaker

Rui Bando
Recruit Lifestyle Co., Ltd.
Data Engineering Unit
Product Manager, Data Planner

Ganbaatar Bya
Recruit Lifestyle Co., Ltd.
Data Engineering Unit
Engineer
Our Company
Our Business

Recruit Matching Platform

A platform that matches Users and Clients

User

Client
Our services
## About our company

### Biz segments

<table>
<thead>
<tr>
<th>HR Technology</th>
<th>HR Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Media &amp; Solutions</strong></td>
<td><strong>HR Technology</strong></td>
</tr>
<tr>
<td>Marketing Solutions</td>
<td>Housing and Real Estate</td>
</tr>
<tr>
<td>Bridal</td>
<td>RSC</td>
</tr>
<tr>
<td>Travel</td>
<td>RMP</td>
</tr>
<tr>
<td>Dining</td>
<td>RLS</td>
</tr>
<tr>
<td>Beauty</td>
<td></td>
</tr>
<tr>
<td><strong>HR Solutions</strong></td>
<td>Recruiting in Japan</td>
</tr>
<tr>
<td><strong>Recruiting in Japan</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Recruiting in Japan</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Overseas Operations</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Trading Volume March 2019

Approx. 180 Billion JPY

### Data Engineering Unit
Crossing dept. that develop and operate data
Data Engineering Unit’s mission

Biz A
Data operation A
Cloud Solutions
Google Cloud Platform

Biz B
Data operation B
Analytics Solutions

Biz C
Data operation C

Cloud & Big data dev & ops
Our Data Solutions

- Item recommendation
- Point distribution optimisation
- Preference estimation
- Sort items
- Real time offer

Revenue Simulation

- RFM / LTV model
- TV-CM effect estimate
- Separation prevention

User

Client

Chat bot

Document and image classification

Price optimization

Plan recommendation

Etc.
Our action
Large-scale services require reliability through stable operation
※Great impact when a problem occurs
Change Development Process

Process improvement including Agile in “Our service”

Architecture is also On-Premises

Existing architectures face difficult challenges
Cloud and our architecture

Focus on new architecture such as Cloud and serverless

- On-Premises to Cloud
- New architecture such as serverless has appeared

Migrating large services is not easy

- Understanding migration is really a challenge.
- Migration cost is comparable with new cloud development services.
- The direction of product is not affected by the keyword “Cloud”

Understand what our users really want!!
For Value Provision

The importance of “Approach method” is increasing
「P: people」「P: process」「T: technology」
Understand these keywords.

Focus on providing valuable services to users
Understand importance of 「How」

Product value is more important to user than architecture
※Don’t be the developer’s self-satisfaction
Anti-pattern: Unbalanced P-P-T

A momentary architecture and development process do not lead to sustainable value provision.
Utilizing tech that prioritizes value provision

No growth of product just by “Protection”

Actively adopt new technologies such as cloud
Maximize value provision to users

Our Service × New Technology

Hybrid efforts in architecture, dev process and operation.

Understand what is needed now without considering new tech as the top priority
Separation of “Agility” and “Reliability”

The important thing is "Give value to user first"

Think about what you should prioritize.
Work on carving out the architecture by considering on
an “Agility” and “Reliability” system

Bimodal IT (SoR/SoE)

Reliability
Waterfall, V-model
Plan-driven, approval-based
Long(months)

Agility
Agile, Kanban
continuous, process-based
Short(weeks, days)
Adopting P-P-T balanced architecture

The adoption of optimal tech raise the level of P-P-T

Balance

People  Process  Technology

Adopt appropriate architecture

People  Process  Technology

Adopt appropriate architecture
Our challenge for user

Always thinking about “How” to realize value provision

- Users do not reach without proper “How”
- Depending on your “How”, the speed to reach users will vary greatly

Awareness of user and business value.
In architecture, development process and operation,
Think of “Reliability” and “Agility” separately,
Not aiming to adopt all new programing languages and architectures

Always thinking to pursue user value provision
Dev. Side
Today’s Message

Product Mission

Focus on users
From developer viewpoint

Reduce development concerns
Our Strategies
MVP development

Q: How can we execute many testings for users?

A: Just quickly develop features & test it.

No. There are too many concerns!
Difficulties of MVP in our company

Rapid Delivery + Existing Tech Debt = Expected Quality + High Reliability

Our company
Strategy 1-1: Separated Environment

✓ independant on tech. debt
✓ increase # of testings
Strategy 1-1: Separated Environment

- Rapid Delivery
- Existing Tech Debt
- Expected Quality
- High Reliability

Our company
Strategy 1-2: Independent Dev Team

PM

Eng

Lead Engineer

Frontend

Backend

Infra

Design

QA
Strategy 1-2: Independent Dev Team

- Rapid Delivery
- Existing Tech Debt
- Expected Quality
- High Reliability

Our company
Failures in Architecture

Tech Stack

- MVP
- Golang
- Nuxt.js
- Microservice

Difficulties

- Monitoring
- Member assign
- Versioning up
- Googlability
Failures in Architecture

Is it MVP???

Tech Stack
- MVP
- Golang
- Nuxt.js
- Microservice

Difficulties
- Monitoring
- Member assign
- Versioning up
- Googlability
Strategy 2: Ruby on Rails

<table>
<thead>
<tr>
<th>Tech Stack</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVP</td>
<td>✓ Easy for beginners</td>
</tr>
<tr>
<td></td>
<td>✓ Stable Framework</td>
</tr>
<tr>
<td>Rails</td>
<td>✓ Matured Ecosystems</td>
</tr>
<tr>
<td>Ruby</td>
<td></td>
</tr>
</tbody>
</table>
Strategy 2: Ruby on Rails

- Rapid Delivery
- High Reliability
- Expected Quality
- Existing Tech Debt

Our company
Strategy 3-1: + Cloud

Tech Stack

- AWS & GCP

Benefits

- More Functionality
- More Flexibility
- Managed Monitoring
Strategy 3-2: Kubernetes

Tech Stack

- AWS & GCP
- MVP

Benefits

- ✓ Auto Healing
- ✓ Auto Scaling
- ✓ Rapid Rollback
Strategy 3: Cloud & Kubernetes

- Rapid Delivery
- Existing Tech Debt
- Expected Quality
- High Reliability

Our company
More Reliability
Overview

- Kubernetes Engine
- Cloud Load Balancing
- Cloud CDN
- Cloud Storage
- Cloud SQL
Conventional parts

- Kubernetes Engine
- Cloud Load Balancing
- Cloud CDN
- Cloud Storage
- Cloud SQL
- Vue.js
- Google Cloud Platform
1. Failover
2. DevOps

- Vue.js
- Cloud CDN
- Cloud Storage
- Cloud Load Balancing
- Kubernetes Engine
- Cloud SQL
- RAILS
2. DevOps

- Vue.js
- Kubernetes Engine
- Cloud CDN
- Cloud Storage
- RAILS
- Cloud Load Balancing
- Cloud SQL
- Docker

1. hook
2. test & build
3. migrate
4. deploy
5. upload

GitHub Enterprise
More Quality
Monitoring

app logs

performance metrics

Cloud Load Balancing

Kubernetes Engine

Cloud SQL

error log filter

alert

slack
1. Easy Error Log explorer
2. Codified Alerts

- To manage app alerts, codified settings in Terraform
  - sharable knowledge
  - less human error: not to let errors be ignored
Challenges
k8s pod lifecycle: Rolling Update

1. Deploy v2
2. v2 pod added
3. once v2 is ready, v1 dies
   ~repeat 2 to 3~
4. all replaced
Rails architecture on k8s

Pod

Nginx

Rails (Puma)

Sidecar (monitoring)

Other pods
k8s pod lifecycle

1. Processes running
2. Receives SIGTERM
3. Each process dies
4. Pod dies
By default, shutdown is not ordered
  ○ Puma start Graceful Shutdown
  ○ Nginx immediately dies
  ○ connection badly closed X(

Graceful Shutdown Problem
Graceful Shutdown Problem: How To Order

1. Puma & Nginx postpones SIGTERM
   a. preStop
2. Start GS in Nginx
   a. till wait puma process ends
3. Start GS in Puma
   a. connections stay
4. Requests gone, receives SIGTERM
   a. Nginx starts shutdown
5. Pod dies
Achievements
More Productivity

175% productivity since previous architecture
Fast Velocity

velocity since previous architecture

30 releases in first 3 months

250%
CVR rift

by UX improvements

110%
Wrap up
TL; DR

To focus on users,

Ruby on Rails on Kubernetes on Clouds
fin.