5 years of Cloud Foundry at Rakuten

Ronak Banka
Carlo Alberto Ferraris
Ronak Banka
@ronakbanka

Carlo Alberto Ferraris
@cafxx
Rakuten

13k+ employee, +3k engineers, 40+ subsidiaries
Cloud Foundry in Rakuten

Past: RPaaS (v1) history, lessons learned

Present: Designing and deploying v2

The future
Cloud Foundry in Rakuten

On-premise PaaS for Rakuten engineers

 Started in 2011, forked in ~2012
Cloud Foundry in Rakuten

Organizations and Crowd integration
Flexible routing (think mod_rewrite)
Application log collection, storage and retrieval (fluentd)
Application monitoring/alerting
Cloud Foundry in Rakuten

CF & RPaaS History

CF v1
- bug fixes
- small improvements
- Fluentd logging Integration
- (Failed PR)

CF v2
- DEA/Warden Hooks support
- Failed to upgrade v2 without downtime.

RPaaS
- Atlassian Crowd Integration
- Flexible Routing (like mod_rewrite)
- Clustered Redis Support
- Clustrix as a Service Support
Cloud Foundry in Rakuten

5000+ vSphere VMs (single-tenant)
1200+ apps
2 independent datacenters
7 engineers
Lessons learned

Know when to say “no”

Not everything is a good fit for PaaS: if it doesn’t fit – don’t force it
Lessons learned

Forking is a very bad idea

(either build on top or on the side)
Lessons learned

Engineers’ time does not scale

(your provisioning* system must be able to)

(chef-solo doesn’t)
Lessons learned

Centralized (logg|monitor|alert)ing is vital

(as long as these systems can scale as well)
(and you have fallbacks)
Lessons learned

Design for failure

(and set expectations accordingly)
Cloud Foundry v2 in Rakuten

Started planning in 2015, production early 2016

On-premises (but plans for hybrid)

Application migration is ongoing
Cloud Foundry v2 in Rakuten

Provisioning/Delivery → Bosh+Concourse

Infrastructure → vSphere+Openstack
Cloud Foundry v2 in Rakuten

Living on multiple clouds
Cloud Foundry v2 in Rakuten

Single manifest → Two intermediate manifests

Properties are shared among manifests
Target individually & deploy
Cloud Foundry v2 in Rakuten
Cloud Foundry v2 in Rakuten

Stack Hack

Before: 3 API endpoints $\rightarrow$ 3 CF deployments (dev/stg/pro)
Now: single API endpoint $\rightarrow$ 3 segregated networks

```yaml
- name: dea_development
  properties:
    dea_next:
      stacks:
      - name: development
        package_path: /var/vcap/packages/rootfs_cflinuxfs2/rootfs
        resource_pool: dea_development_pool
```
Cloud Foundry v2 in Rakuten

How we deploy

Sync with upstream
CI for in-house bosh releases & CF plugins
Monitor live telemetry as we deploy
Serverspec for individual components
Infratester for subsystems
Acceptance & smoke tests
Cloud Foundry v2 in Rakuten
Cloud Foundry v2 in Rakuten

Monitor everything

System metrics
Component metrics
External metrics
End-to-end metrics (passive/active)
Cloud Foundry v2 in Rakuten

Watchtower: end-to-end active monitoring
Cloud Foundry v2 in Rakuten

Rakuten-specific features and integrations are implemented in our API and in our CLI plugins

Organization bootstrapping
Billing
Service brokers data feeds
Log access
Cloud Foundry v2 in Rakuten

Know when to say “no”
Forking is a bad idea
Engineers’ time does not scale
Centralized (logg|monitor|alert)ing is vital
Design for failure
Cloud Foundry vNext in Rakuten

Azure deployments to enable burst-to-cloud
Integration with service providers (Azure, Trove)
HTTP/2, SSL cert auto-provisioning
Autoscaling
DC federation
Dear Santa...

PRs – fast turnaround, document the conventions
Standards – sysctl, cronjobs, rsyslog (colocation)
Dry-run – bosh deploy and job template preview

Log – Multiline, reliably know we’re missing logs
Hooks – validate (or notify on) API actions
Q&A

@ronakbanka @cafxx