Large-scaled Deploy Over 100 Servers in 3 Minutes

Deployment strategy for next generation

GMOペパボ株式会社



你好朋友!

self.introduce



=>
{
  name: “SHIBATA Hiroshi”,
  nickname: “hsbt”,
  title: “Chief engineer at GMO Pepabo, Inc.”,
  commit_bits: [“ruby”, “rake”, “rubygems”, “rdoc”, “psych”, “ruby-build”, “railsgirls”, “railsgirls-jp”],
  sites: [“www.ruby-lang.org”, “bugs.ruby-lang.org”, “rubyci.com”, “railsgirls.com”, “railsgirls.jp”],
}

I’m from Asakusa.rb

Asakusa.rb is one of the most active meet-ups in Tokyo, Japan.

@a_matsuda (Ruby/Rails committer, RubyKaigi chief organizer)
@kakutani (RubyKaigi organizer)
@ko1 (Ruby committer)
@takkanm (Ruby/Rails programmer)
@hsbt (Me!)

and many Rubyists in Japan.



GMOペパボ株式会社



最新ニュース | セミナー・イベント情報 | IRニュース

2015.08.04 講演・出演情報 GMOペパボ

「YAPC::Asia 2015」に登壇します

グローバルサイン認証サイト
SSL secured
クリックして確認
GlobalSign by GMO

株価情報



RubyKaigi 2016

at Kyoto International Conference Center, Kyoto, Japan
Sep. 8th - 10th

› Call for Speakers › Registration › Call for Sponsors

› Anti Harassment Policy

For news and updates, follow @rubykaigi on Twitter

Past Kaigis | 2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2009 | 2008 | 2007 | 2006

This work by RubyKaigi 2016 Team is licensed under a Creative Commons Attribution 3.0 Unported License.

Call for Speakers

RubyKaigi 2016 open

RubyKaigi 2016 is an international Ruby conference that will take place in Kyoto, Japan from September 8th to September 10th 2016, and we're looking for speakers!

Submit a proposal

Closes at **May 22, 2016 at 12:00am JST**

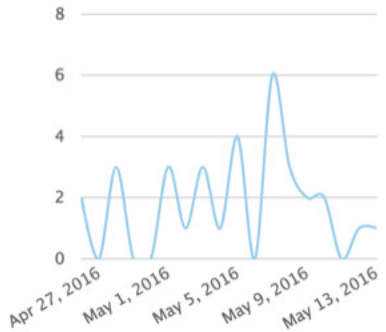
9 days left to submit your proposal!

Guidelines

- Your topic should be related to the Ruby programming language
- Taking a look at "Past Kaigis" (2015, 2014, 2013) linked from http://rubykaigi.org/2016 would give you an idea of the taste of the Kaigi
- Your talk slot will be about 40-45 minutes
- Your talk can be spoken in either English or Japanese. All Japanese talks will be simultaneously interpreted to English
- The CFP form will remain open until May 22nd
- You need a GitHub and/or Twitter account to submit your proposal via this submission form
- We recommend you to submit your proposal early. The earlier you submit, the better chance for your CFP to be reviewed and get feedback from the reviewers online
- We require all selected talks to comply wholly with our Anti-Harassment Policy

CFP Stats

32 proposals



About the conference ticket

- Accepted speakers will get free admission to the conference
- Even if your talk is not accepted, you can get your RubyKaigi ticket at "Super Early Bird" price if you submit a proposal

GMOペパボ株式会社

Deployment Strategy for Next Generation

2014/11/xx

CEO and CTO said…

CEO: “We are going to promote our service on TV CM! at Feb, 2015”

CTO: “Do make out service to scalable, redundant, high-performance architecture! in 3 months”

Me: “Yes, I do it!!1”

Our service status at 2014/11

It’s simply Rails Application with IaaS (not Heroku)

- 6 application servers
- To use capistrano 2 for deployment
- Mixed background job, application processes and batch tasks



Our service issue

Do scale-out

Do scale-out with automation!

Do scale-out with rapid automation!!

Do scale-out with extremely rapid automation!!!

Do scale-out with automation

Concerns of bootstrap instructions

Typical scenario of server set-up for scale out.

- OS boot
- OS Configuration
- Provisioning with puppet/chef
- Setting up to capistrano
- Deploy rails application
- QA Testing
- Added load balancer (= Service in)

Web operation is manual instructions

- We have been created OS Image called “Golden Image” from running server
- Web operations such as os configuration and instances launch are manual instruction.
- Working time is about 4-6 hours

- It’s blocker for scale-out largely.

No ssh

We added “No SSH” into our rule of Web operation



Background of “No SSH”

In large scale service, 1 instance is like a “1 process” in Unix environments.

We didn’t attach process using gdb usually.

- We don’t access instance via ssh

We didn’t modify program variables in memory usually.

- We don’t modify configuration on instance

We can handle instance/process status using api/signal only.

puppet

Provision with puppet

We have puppet manifests for provision. but It's sandbox status.

- It based on old Scientific Linux
- Some manifest is broken…
- Service developers didn't use puppet for production

At first, We fixed all of manifests and enabled to deploy to production environments.

```
% ls **/*.pp | xargs wc -l | tail -1
  5546 total
```

To use puppetmasterd

- We choice master/agent model
- It's large scaled architecture because we didn't need to deploy puppet manifests each servers.
- We already have puppetmasterd manifests written by puppet using passenger named rails application server.

https://docs.puppetlabs.com/guides/passenger.html

cloud-init

What's cloud-init

"Cloud-init is the defacto multi-distribution package that handles early initialization of a cloud instance."

https://cloudinit.readthedocs.org/en/latest/

- We(and you) already used cloud-init for customizing to configuration of OS at initialization process on IaaS
- It has few documents for our use-case…

Basic usage of cloud-init

We only use OS configuration. Do not use “run_cmd” section.

```
#cloud-config
repo_update: true
repo_upgrade: none

packages:
 - git
 - curl
 - unzip

users:
  - default

locale: ja_JP.UTF-8
timezone: Asia/Tokyo
```

Image creation with itself

We use IaaS API for image creation with cloud-init userdata.

```
puppet agent -t

rm -rf /var/lib/cloud/sem /var/lib/cloud/instances/*

aws ec2 create-image --instance-id `cat /var/lib/cloud/data/instance-id` --name
www_base_`date +%Y%m%d%H%M`
```

We can create OS Image using cloud-init and provisioned puppet when boot time of instance.

Do scale-out with rapid automation

Upgrade Rails app

Upgrading Rails 4

- I am very good at “Rails Upgrading”
- Deploying in Production was performed with my colleague named @amacou

```
% g show c1d698e
commit c1d698ec444df1c137a301e01f59e659593ecf76
Author: amacou <amacou.abf@gmail.com>
Date:   Mon Dec 15 18:22:34 2014 +0900

    Revert "Revert "Revert "Revert "[WIP] Rails 4.1.X へのアップグレード""""
```

What's new for capistrano3

"A remote server automation and deployment tool written in Ruby."

http://capistranorb.com/

Example of Capfile:

```
require 'capistrano/bundler'
require 'capistrano/rails/assets'
require 'capistrano3/unicorn'
require 'capistrano/banner'
require 'capistrano/npm'
require 'slackistrano'
```

We rewrite own capstrano2 tasks to capistrano3 convention

Do not use hostname/ip dependency

We discarded dependencies of hostname and ip address.

```
config.ru:
10:  defaults = `hostname`.start_with?('job') ?

config/database.yml:
37:  if `hostname`.start_with?(‘search')

config/unicorn.conf:
6:  if `hostname`.start_with?('job')
```

Use API of IaaS for our use-case.

Rails bundle

Bundled package of Rails application

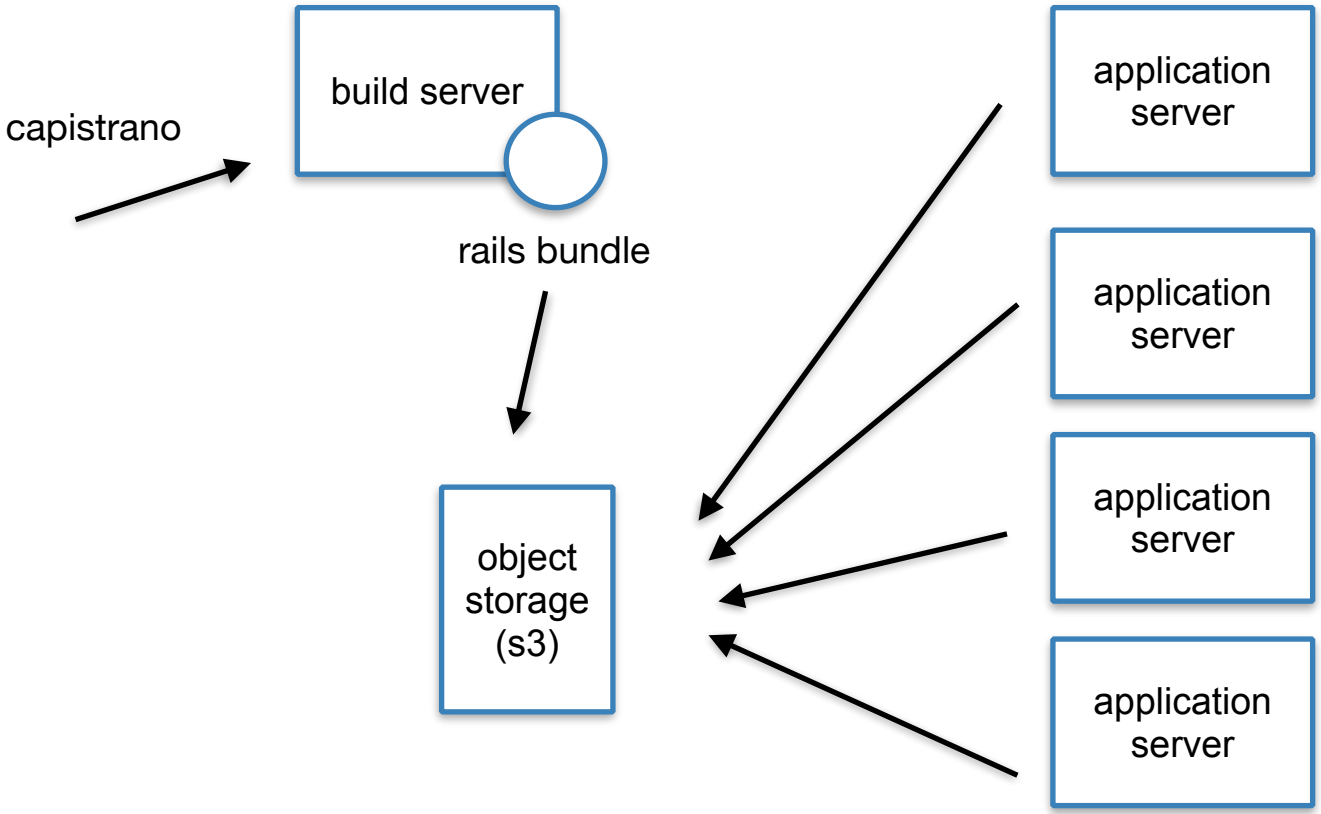
Prepared to standalone Rails application with rubygems and precompiled assets

```
$ bundle exec cap production archive_project ROLES=build
```

Part of capistrano tasks:

```
desc "Create a tarball that is set up for deploy"
task :archive_project =>
  [:ensure_directories, :checkout_local, :bundle, :npm_install, :bower_install,
   :asset_precompile, :create_tarball, :upload_tarball, :cleanup_dirs]
```

Distributed rails package



Integration of image creation

We extracted rails bundle when instance creates self image with clout-init.

```
# Fetch latest application package
RELEASE=`date +%Y%m%d%H%M`
ARCHIVE_ROOT='s3://rails-application-bundle/production/'
ARCHIVE_FILE=$(
  aws s3 ls $ARCHIVE_ROOT | grep -E 'application-.*.tgz' | awk '{print $4}' | sort -r | head -n1
)
aws s3 cp "${ARCHIVE_ROOT}${ARCHIVE_FILE}" /tmp/rails-application.tar.gz

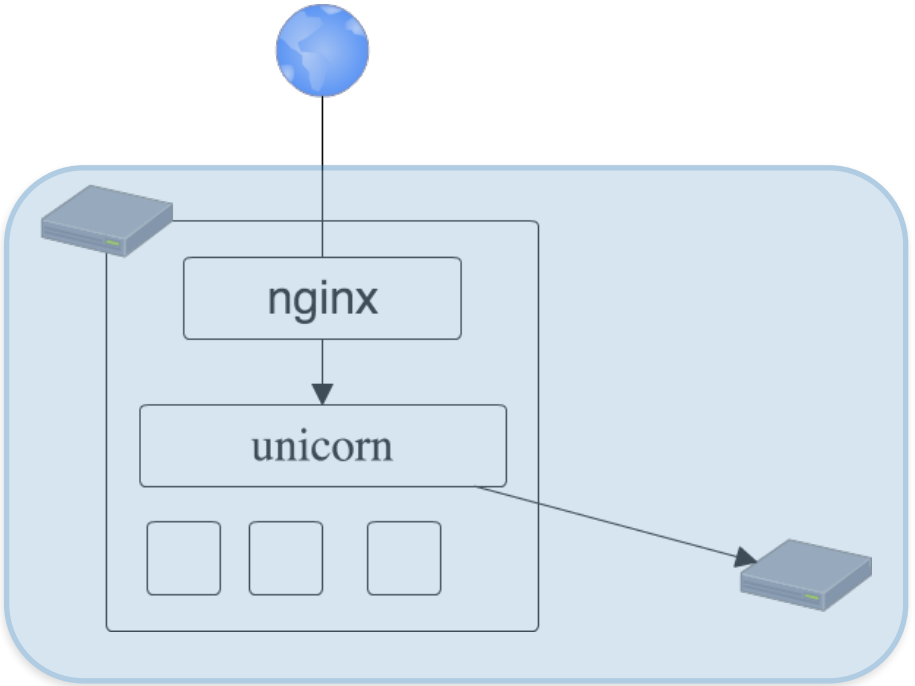
# Create Directories of capistrano convention
(snip)

# Invoke to chown
(snip)
```

How to test instance behavior

We need to guarantee http status from instance response.

We removed package version control from our concerns.



thor

What's thor

"Thor is a toolkit for building powerful command-line interfaces. It is used in Bundler, Vagrant, Rails and others."

http://whatisthor.com/

```
module AwesomeTool
  class Cli < Thor
    class_option :verbose, type: :boolean, default: false

    desc 'instances [COMMAND]', 'Desc'
    subcommand('instances', Instances)
  end
end
```

```
module AwesomeTool
  class Instances < Thor
    desc 'launch', 'Desc'
    method_option :count, type: :numeric, aliases: "-c", default: 1
    def launch
      (snip)
    end
  end
end
```

Scale out with cli command

We can scale out with one command via our cli tool

```
$ some_cli_tool instances launch -c …
$ some_cli_tool mackerel fixrole
$ some_cli_tool scale up
$ some_cli_tool deploy blue-green
```

All of web operations should be implement by command line tools

How to automate instructions

- Write real-world instructions
- Pick instruction for automation
- DO automation

Do scale-out with extremely rapid automation

Concerns of bootstrap time

Typical scenario of server set-up for scale out.

- OS boot
- OS Configuration
- Provisioning with puppet/chef
- Setting up to capistrano
- Deploy rails application
- Added load balancer (= Service in)

We need to enhance to bootstrap time extremely.

Concerns of bootstrap time

Slow operation

- OS boot
- Provisioning with puppet/chef
- Deploy rails application

Fast operation

- OS Configuration
- Setting up to capistrano
- Added load balancer (= Service in)

Check point of Image creation

Slow operation

- OS boot

Step1

- Provisioning with puppet/chef

Step2

- Deploy rails application

Fast operation

- OS Configuration
- Setting up to capistrano
- Added load balancer (= Service in)

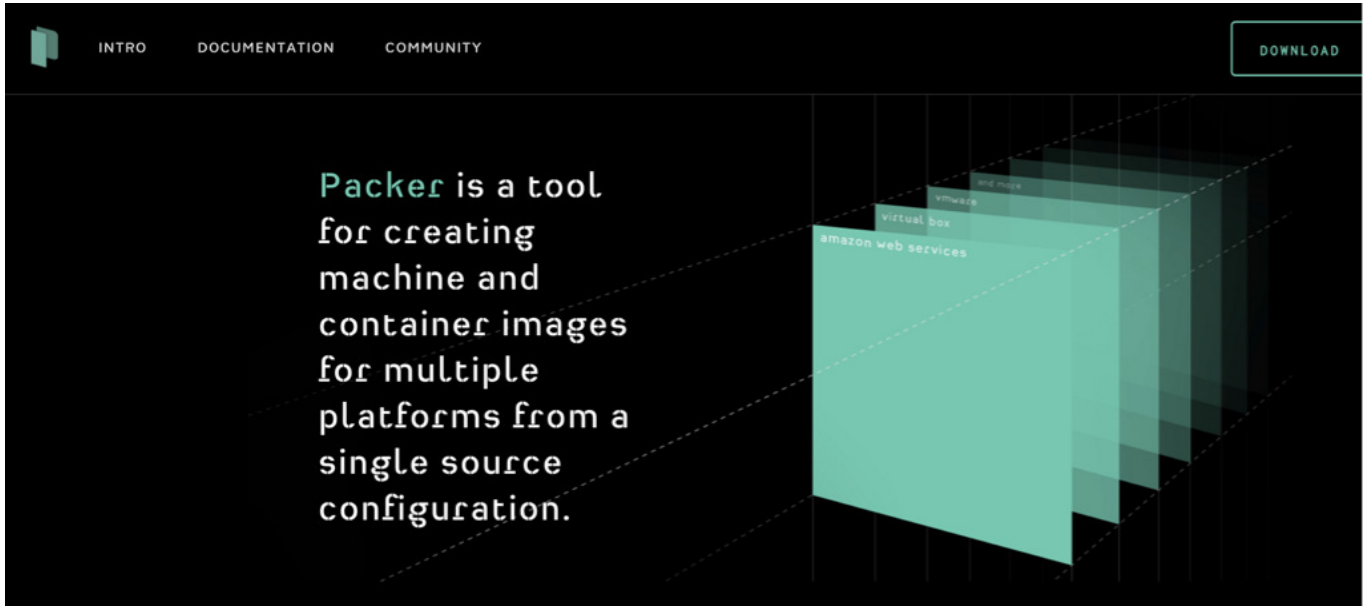
2 phase strategy

- Official OS image
  - Provided from platform like AWS, Azure, GCP, OpenStack…
- Minimal image(phase 1)
  - Network, User, Package configuration
  - Installed puppet/chef and platform cli-tools.
- Role specified(phase 2)
  - Only boot OS and Rails application

Packer

Use-case of Packer

I couldn’t understand use-case of packer. Is it Provision tool? Deployment tool?



inside image creation with Packer

- Packer configuration
  - JSON format
  - select instance size, block volume
- cloud-init
  - Basic configuration of OS
  - only default module of cloud-init
- provisioner
  - shell script :)
- Image creation
  - via IaaS API

minimal image

cloud-init

```
#cloud-config
repo_update: true
repo_upgrade: none

packages:
 - git
 - curl
 - unzip

users:
  - default

locale: ja_JP.UTF-8
timezone: Asia/Tokyo
```

provisioner

```
rpm -ivh http://yum.puppetlabs.com/
puppetlabs-release-el-7.noarch.rpm

yum -y update
yum -y install puppet
yum -y install python-pip
pip install awscli

sed -i 's/name: centos/name: cloud-user/' /etc/
cloud/cloud.cfg
echo 'preserve_hostname: true' >> /etc/cloud/
cloud.cfg
```

web application image

cloud-init

```
#cloud-config
preserve_hostname: false
```

provisioner

```
puppet agent -t

# Fetch latest rails application
(snip)

# enabled cloud-init again
rm -rf /var/lib/cloud/sem /var/lib/cloud/instances/*
```

Integration tests with Packer

We can tests results of Packer running. (Impl by @udzura)

packer configuration

```
 "provisioners": [
(snip)
 {
    "type": "shell",
    "script": "{{user `project_root`}}packer/minimal/provisioners/run-serverspec.sh",
    "execute_command": "{{ .Vars }} sudo -E sh '{{ .Path }}'"
  }
]
```

run-serverspec.sh

```
yum -y -q install rubygem-bundler
cd /tmp/serverspec
bundle install --path vendor/bundle
bundle exec rake spec
```

We created cli tool with thor

We can run packer over thor code with advanced options.

```
module SomeCliTool
  class Ami < Thor
    method_option :ami_id,        type: :string,  aliases: "-a"
    method_option :init,          type: :boolean
    desc 'build-www', 'wwwの最新イメージをビルドします'
    def build_www
      …
    end
  end
end
```

```
$ some_cli_tool ami build-minimal
$ some_cli_tool ami build-www
$ some_cli_tool ami build-www —init
$ some_cli_tool ami build-www -a ami-id
```

Scale-out Everything

What’s blocker for scale-out

- Depends on manual instruction of human
- Depends on hostname or ip address architecture and tool
- Depends on persistent server or workflow like periodical jobs
- Depends on persistent storage

consul

Nagios

We used nagios for monitoring to service and instance status.

But we have following issue:

- nagios don’t support dynamic scaled architecture
- Complex syntax and configuration

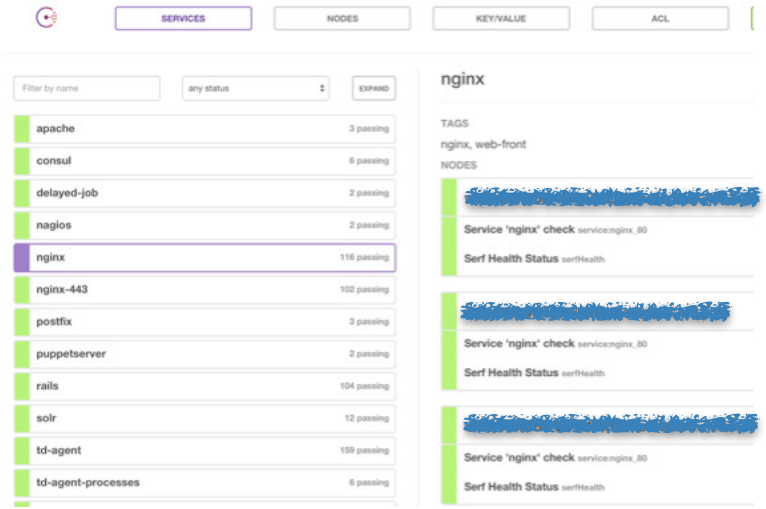
We decided to remove nagios for service monitoring.

consul + consul-alert

We use consul and consul-alerts for process monitoring.

https://github.com/hashicorp/consul
https://github.com/AcalephStorage/consul-alerts

It provided to discover to new instances automatically and alert mechanism with slack integration.



mackerel

munin

We used munin for resource monitoring

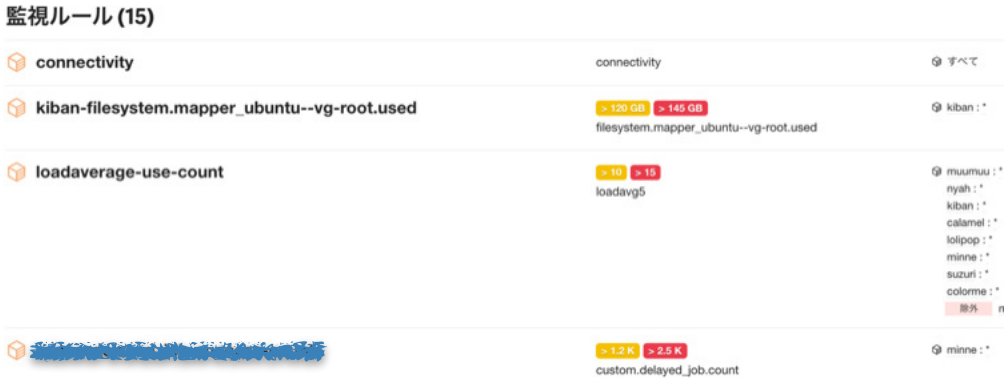
But munin doesn’t support dynamic scaled architecture. We decided to use mackerel.io instead of munin.

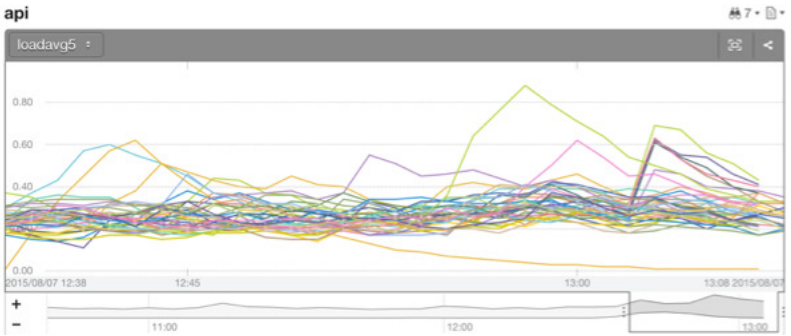


Mackerel

“A Revolutionary New Kind ofApplication Performance Management. Realize the potential in Cloud Computingby managing cloud servers through “roles””

https://mackerel.io





Configuration of mackrel

You can added instance to role(server group) on mackerel with mackerel-agent.conf

```
[user@www ~]$ cat /etc/mackerel-agent/mackerel-agent.conf
apikey = “your_api_key”
role = [ "service:web" ]
```

And You can made your specific plugin for mackerel. It’s simple convention and compatible for munin and nagios.

Many of Japanese developer made useful mackerel plugin written by Go/mruby.

fluentd

access_log aggregator with td-agent

We need to collect access-log of all servers with scale-out.

https://github.com/fluent/fluentd/

We used fluentd to collect and aggregate.

```
<match nginx.**>
  type forward
  send_timeout 60s
  recover_wait 10s
  heartbeat_interval 1s
  phi_threshold 16
  hard_timeout 60s

  <server>
    name aggregate.server
    host aggregate.server
    weight 100
  </server>
 <server>
    name aggregate2.server
    host aggregate2.server
    weight 100
    standby
  </server>
</match>
```

```
<match nginx.access.*>
  type copy

 <store>
   type file
   (snip)
 </store>

 <store>
    type tdlog
    apikey api_key
    auto_create_table true
    database database
    table access
    use_ssl true
    flush_interval 120
    buffer_path /data/tmp/td-agent-td/access
  </store>
</match>
```

Scheduler with sidekiq

Remove to batch scheduler

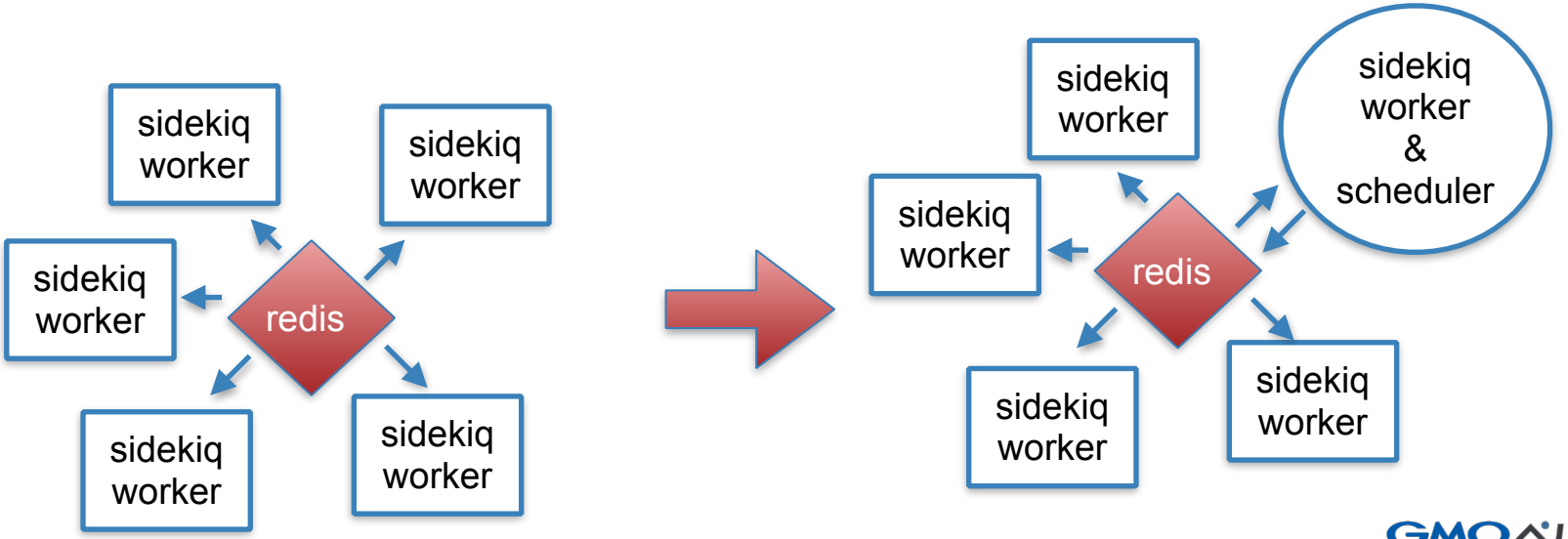
We need to use `batch` role for scheduled rake task. We have to create some payments transaction, send promotion mail, indexing search items and more.

We use `whenever` and cron on persistent state server. but It could not scale-out largely and It's SPOF.

I use sidekiq-scheduler and consul cluster instead of cron for above problems.

scheduler architecture

sidekiq-scheduler (https://github.com/moove-it/sidekiq-scheduler) allows periodical job mechanism to sidekiq server.
We need to specify a enqueue server in sidekiq workers. I elected enqueue server used consul cluster.



Test Everything

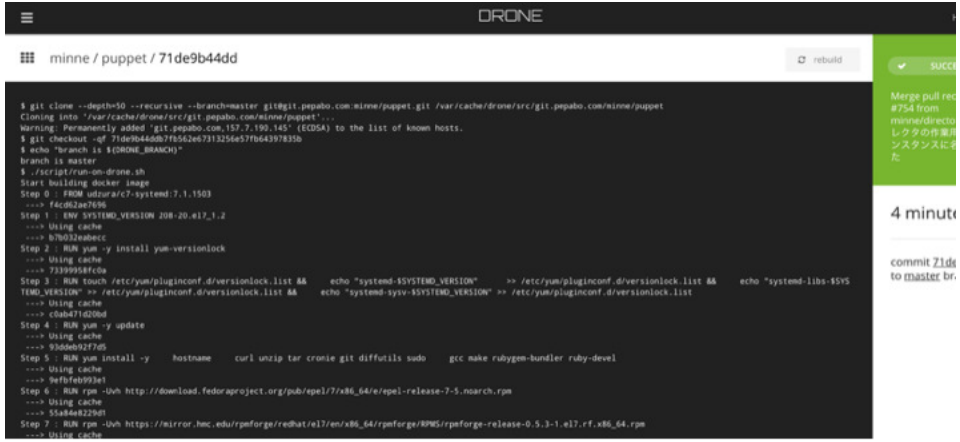
Container CI

Drone CI

“CONTINUOUS INTEGRATION FOR GITHUB AND BITBUCKET THAT MONITORS YOUR CODE FOR BUGS”

https://drone.io/

We use Drone CI on our Openstack platform named “nyah”



Container based CI with Rails

We use Drone CI(based docker) with Rails Application. We need to separate Rails stack to following containers.

- rails(ruby and nodejs)
- redis
- mysql
- elasticsearch

And We invoke concurrent test processes used by test-queue and teaspoon.

Infra CI

What's Infra CI

We test server status such as lists of installed packages, running processes and configuration details continuously.

Puppet + Drone CI(with Docker) + Serverspec = WIN

We can refactoring puppet manifests aggressively.

Serverspec

“RSpec tests for your servers configured by CFEngine, Puppet, Ansible, Itamae or anything else.”

http://serverspec.org/

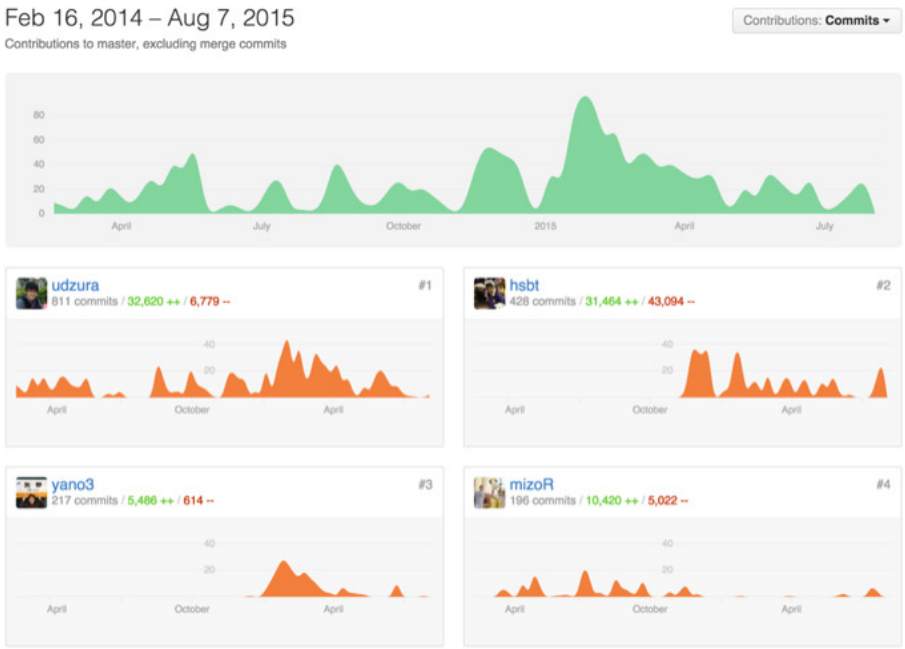
```
% rake -T
rake mtest              # Run mruby-mtest
rake spec               # Run serverspec code for all
rake spec:base          # Run serverspec code for base.minne.pbdev
rake spec:batch         # Run serverspec code for batch.minne.pbdev
rake spec:db:master     # Run serverspec code for master db
rake spec:db:slave      # Run serverspec code for slave db
rake spec:gateway       # Run serverspec code for gateway.minne.pbdev
(snip)
```

Refactoring puppet manifets

We replaced “puppetserver” written by Clojure.

We enabled future-parser. We fixed all of warnings and syntax error.

We added and removed manifests everyday.



Switch Scientific Linux 6 to CentOS 7

We can refactoring to puppet manifests with infra CI.

We added case-condition for SL6 and Centos7

```
if $::operatingsystemmajrelease >= 6 {
  $curl_devel = 'libcurl-devel'
} else {
  $curl_devel = 'curl-devel'
}
```

All of processes under the systemd

We have been used daemontools or supervisord to run background processes.

These tools are friendly for programmer. but we need to wait to invoke their process before invoking our application processes like unicorn, sidekiq and other processes.

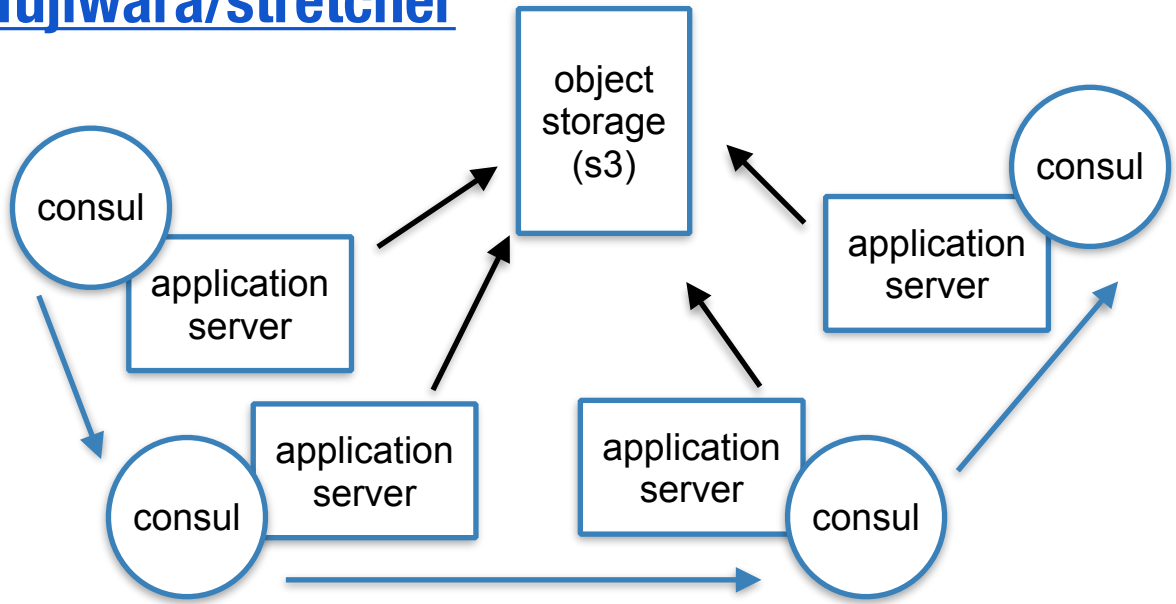
We use systemd for invoke to our application processes directly. It’s simple syntax and fast.

Pull strategy Deployment

stretcher

“A deployment tool with Consul / Serf event.”

https://github.com/fujiwara/stretcher



capistrano-strecher

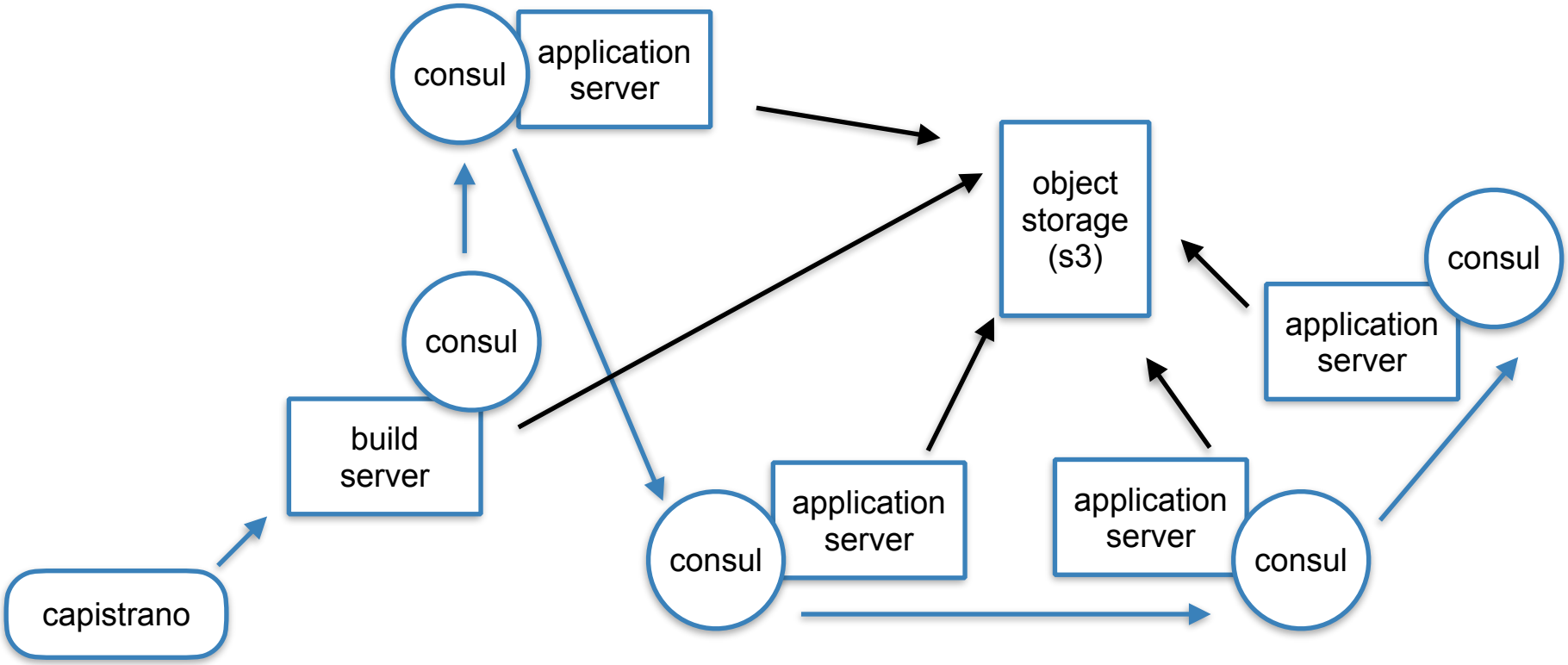
It provides following tasks for pull strategy deployment.

- Create archive file contained Rails bundle
- Put archive file to blob storage like s3
- Invoke consul event each stages and roles

You can use pull strategy deployment easily by capistrano-stretcher.

https://github.com/pepabo/capistrano-stretcher

Architecture of pull strategy deployments



OpenStack

Why we choose OpenStack?

OpenStack is widely used big company like Yahoo!Japan, DeNA and NTT Group in Japan.

We need to reduce running cost of IaaS. We tried to build OpenStack environment on our bare-metal servers.

(snip)

Finally, We've done to cut running cost by 50%

yaocloud and tool integration

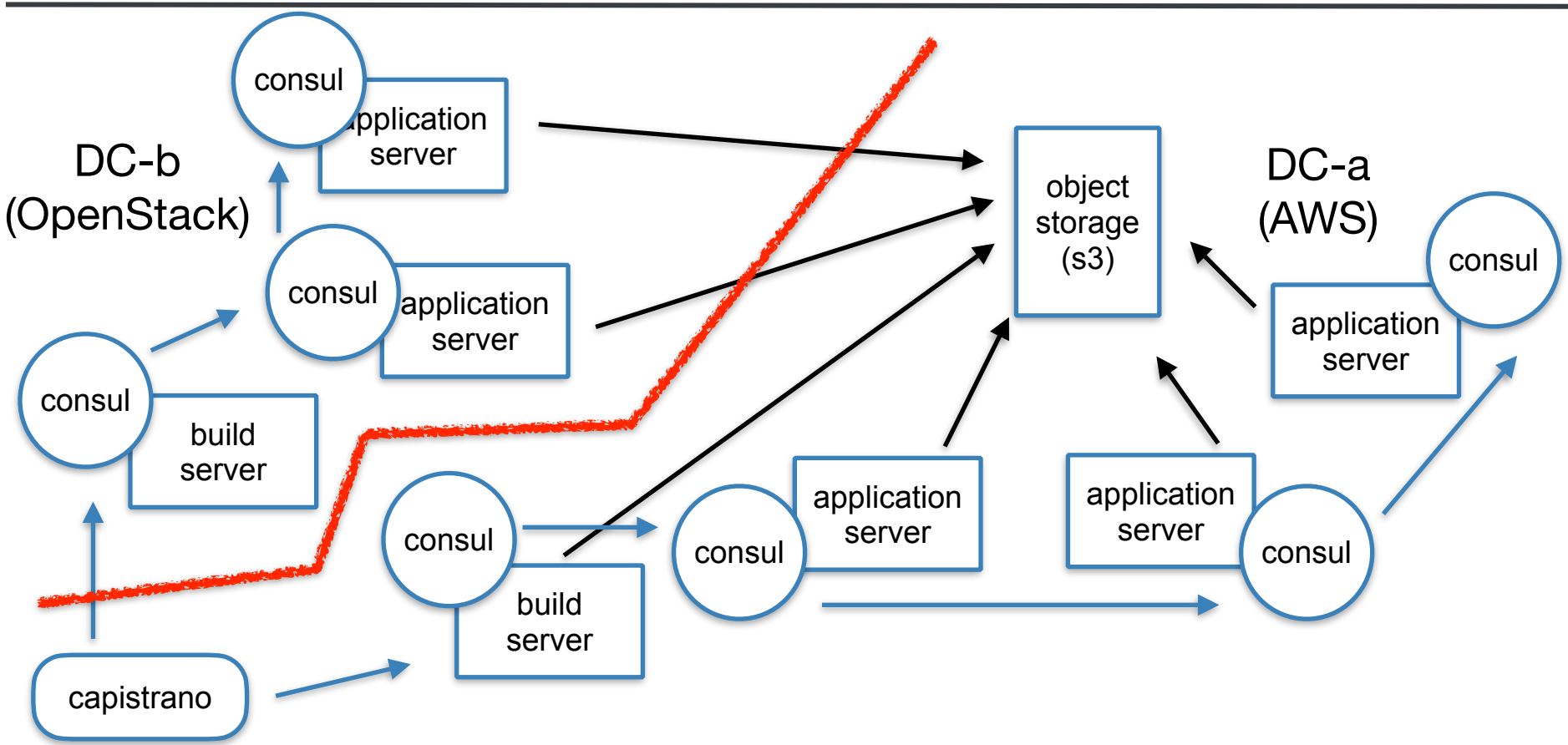
We made Ruby client for OpenStack named Yao.

https://github.com/yaocloud/yao

```
$ Yao::Tenant.list
$ Yao::SecurityGroup.list
$ Yao::User.create(name: name, email: email, password: password)
$ Yao::Role.grant(role_name, to: user_hash["name"], on: tenant_name)
```

It likes aws-sdk on AWS. We can manipulate compute resource using ruby with Yao.

Multi DC deployments in 3 minutes



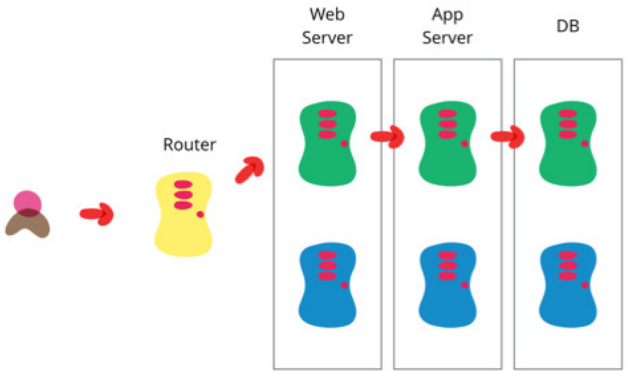
Blue-Green Deployment

Instructions of Blue-Green deployment

Basic concept is following instructions.

1. Launch instances using OS imaged created from Packer
2. Wait to change “InService” status
3. Terminate old instances

That’s all!!1



http://martinfowler.com/bliki/BlueGreenDeployment.html

Dynamic upstream with load balancer

ELB

- Provided by AWS, It's best choice for B-G deployment
- Can handle only AWS instances

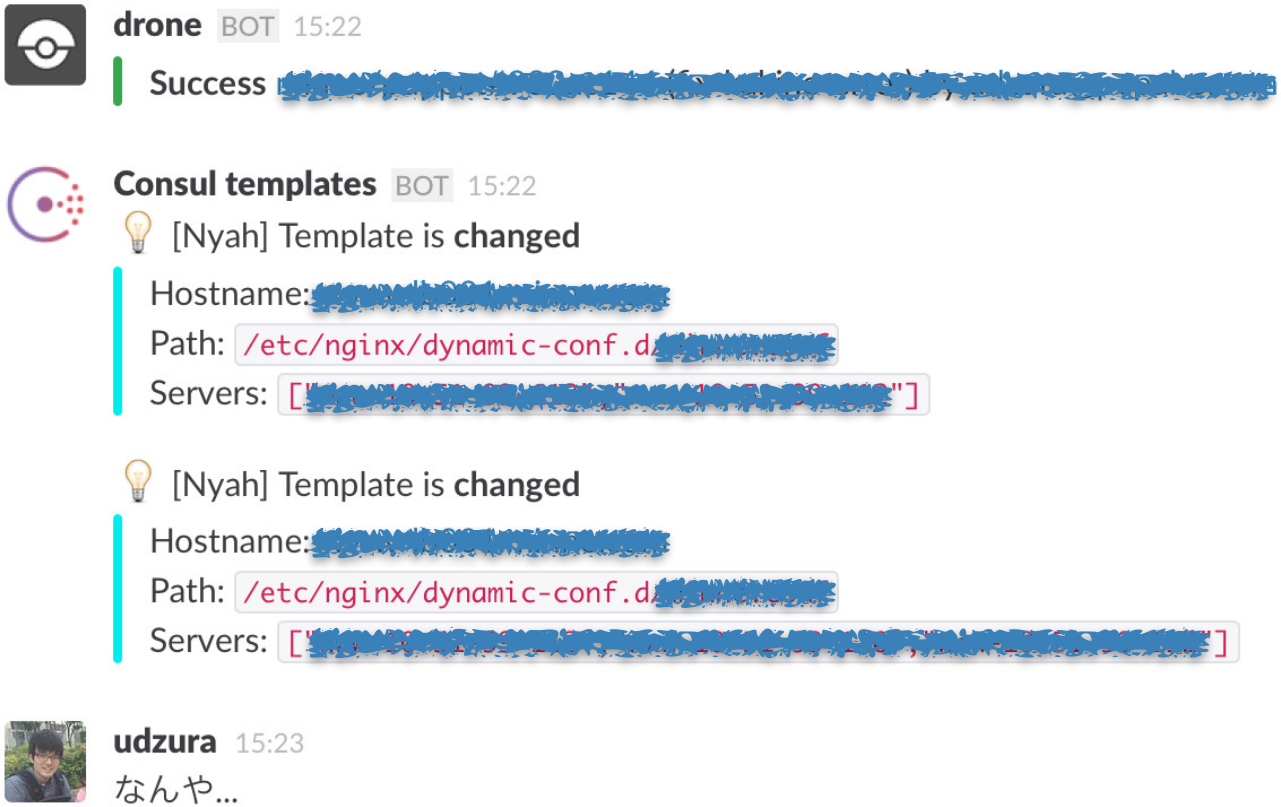
nginx + consul-template

- Change upstream directive used consul and consul-template

ngx_mruby

- Change upstream directive used mruby

Slack integration of consul-template



Example code of thor

```
old_instances = running_instances(load_balancer_name)
invoke Instances, [:launch], options.merge(:count => old_instances.count)

catch(:in_service) do
  sleep_time = 60
  loop do
    instances = running_instances(load_balancer_name)
    throw(:in_service) if (instances.count == old_instances.count * 2) &&
                          instances.all?{|i| i.status == 'InService'}
    sleep sleep_time
    sleep_time = [sleep_time - 10, 10].max
  end
end

old_instances.each do |oi|
  oi.delete
end
```

Summary

- We can handle TV CM and TV Show used by scale-out servers.
- We can enhance infrastructure every day.
- We can deploy rails application over the 100 servers every day.
- We can upgrade OS or Ruby or middleware every day

Yes, We can!