Advanced technic for OS upgrading in 3 minutes

Deployment strategy for next generation

GMOペパボ株式会社



self.introduce



=>
{
  name: “SHIBATA Hiroshi”,
  nickname: “hsbt”,
  title: “Chief engineer at GMO Pepabo, Inc.”,
  commit_bits: [“ruby”, “rake”, “rubygems”, “rdoc”, “tdiary”, “hiki”, “railsgirls”, “railsgirls-jp”, “jenkins”],
  sites: [“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 organizer)
@kakutani (RubyKaigi organizer)
@ko1 (Ruby committer)
@takkanm (Ruby/Rails programmer)
@gunjisatoshi (Rubyist Magazine editor)
@hsbt (Me!)





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

もっと
おもしろく
できる


Mobile
Factory


Wano
Cultivate your dream

スポンサー


ADWAYS


@type


SSRI


HDE
Liberation of Technology


epoch


KAIZEN PLATFORM


面白法人
カヤック


Qiita:Team


Catchball 21
Beyond All Imagination


Gunosy


Kyoto
University
Micro Computer Club


GMOペパボ株式会社


JIG-SAW


Seesaa


six apart


SHANON
for Successful Business.


Speee


diverse

8月
29

第3回ペパボテックカンファレンス

主催：GMOペパボ株式会社



ハッシュタグ： #pbtech

フォロー参加者

募集内容		
	学生枠 無料	先着順(抽選終了) 4/10人
	一般枠 無料	先着順 75/70人

グループ

メンバーです

GMOペパボ株式会社

「もっとおもしろくできる」という企業理念のもと、様々なインターネットサービスを展開しています。

イベント数 4回
メンバー数 399人

開催前

2015/08/29(土)
13:00 ～ 18:00

Googleカレンダー icsファイル

このイベントに申し込む

募集期間
2015/08/07(金) 12:00 ～
2015/08/29(土) 18:00

イベントへのお問い合わせ

会場

2014/11/xx

2014/11/xx …



CTO: antipop

“しばたさん… ちょっといいですか”



“100倍の話しか聞きませんよ”

2014/11/xx …



CTO: antipop

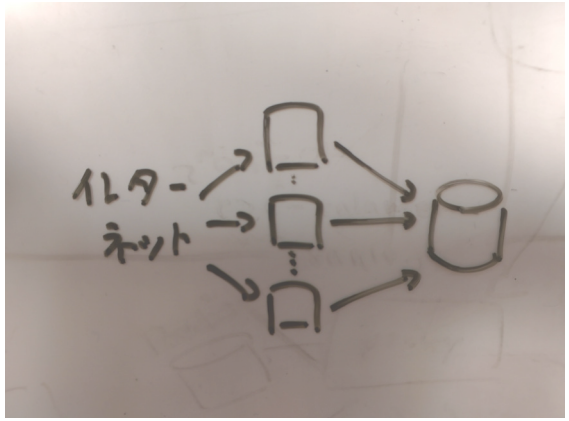
“ある意味 100 倍すね…3ヶ月後に某サービスで CM打つことになりまして…放映までにサービスをバーン!!1としてもらいたいんですよ”



“(うわー)”

Our service status at 2014/11

- Simply Rails Service with IaaS
- 6 application servers
- To use capistrano 2 for deployment
- Mixed worker and application role
- Unknown role server like handled only POST request server



Our service issue

Do scale-out

Do scale-out with automation!

Do scale-out with rapid automation!!!

Do scale-out with extremely rapid automation!!!1

Team member

hsbt: Director, fullstack Programmer

udzura: fullstack Programmer

yano3: fullstack Programmer







GMOペパボ株式会社

Do scale-out

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
- We say it “Tanpopo works…”

- It’s blocker for scale-out largely.

puppet

Fixed all of puppet manifests

- It based on Scientific Linux 6.x
- 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
```

Setting up 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

Use provision tool for scale-out

- Launch instance from raw linux image that it's not customized with our service.
- Deploy rails application with basic instructions.
- Test with single instance
- Attach instance to load balancer



It's puppet work, not tanpopo work

Check Point 0

We need to understand our server configuration via “CODE”

Use provision tool like puppet/chef/ansible etc etc…

Bootstrap time = 4-6 hours

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
- Added load balancer (= Service in)

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 signal/api only.

We have awesome operation tools

- clout-init
- packer
- consul
- IaaS api/cli

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 OS configuration at initialization process on IaaS
- It has few documents for our use-case…

Tuning tools(cloud-init)

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

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

packages:
 - git
 - curl
 - unzip

users:
  - default

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

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?(‘solr')

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

Use API of IaaS for our use-case.

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.

Rails

Upgrading Rails 4

- I am very good at “Rails Upgrading”
- Deploying in Production was performed with @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""""
```



Check point 1

- DO NOT change main architecture
- Write real-world instructions
- Pick instruction for automation
- DO automation

Bootstrap time = 1hours

Do scale-out with rapid automation

capistrano3

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

Rails bundle

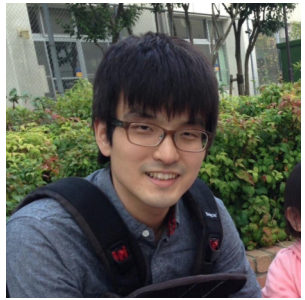
Bundled package of Rails application

Prepared to standalone Rails application with rubygems and precompiled assets

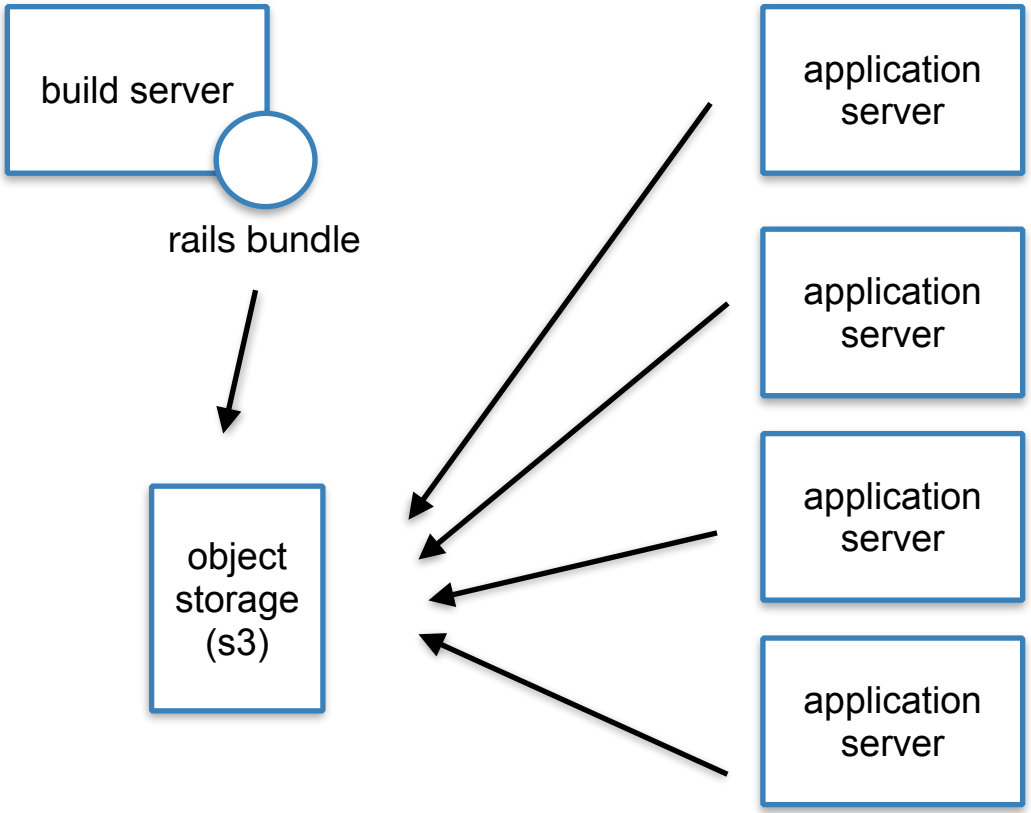
```
$ bundle exec cap production archive_project ROLES=build
```

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.

```
# 最新のアプリケーションの取得
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

# cap setup 相当を実行
(snip)

# chown を実行
(snip)
```

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

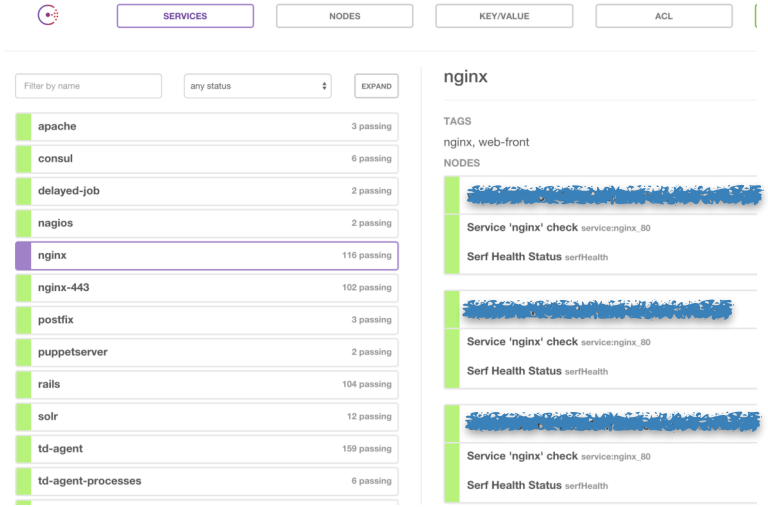
We decided to use nagios for service monitoring like http status with load balancer only.

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

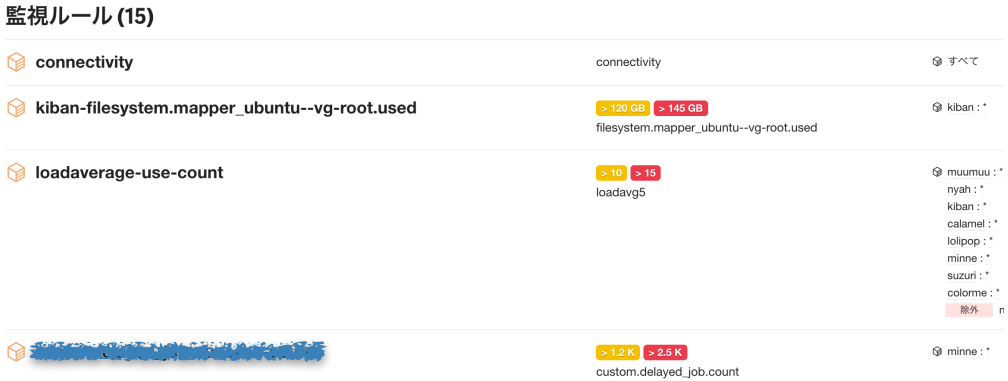
But munin doesn’t support dynamic scaled architecture. We decided to use mackerel.io instead of munin.

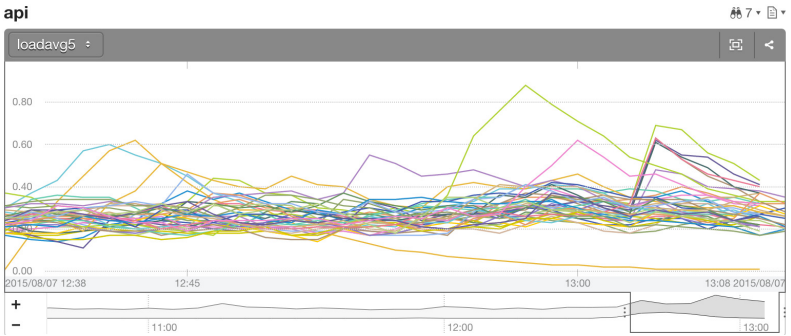


Mackerel

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

https://mackerel.io





GMOペパボ株式会社

Auto join and leave with mackrel

You can added instance to role(server group) on mackerel with mackerel.con

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

You can remove instance from mackerel when instance shutdown. We added following script to initscripts

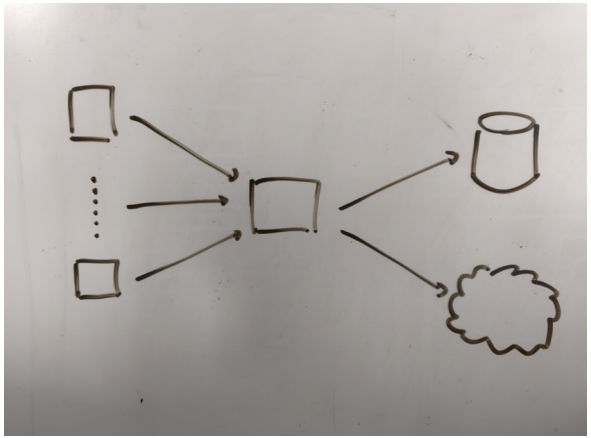
```
curl -s -X POST -H 'Content-type: application/json' -H ‘X-Api-Key:api_key' \
  https://mackerel.io/api/v0/hosts/`cat /var/lib/mackerel-agent/id`/retire
```

※ It’s official support now http://blog-ja.mackerel.io/entry/2015/07/31/105300

fluentd

access_log aggregator with td-agent

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



```
<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>
```

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

Check point 2

- Use cloud-oriented architecture
- Adopt next generation architecture aggressively
- Web operations should be provided from programs

Bootstrap time = 20-30min

GMOペパボ株式会社

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)

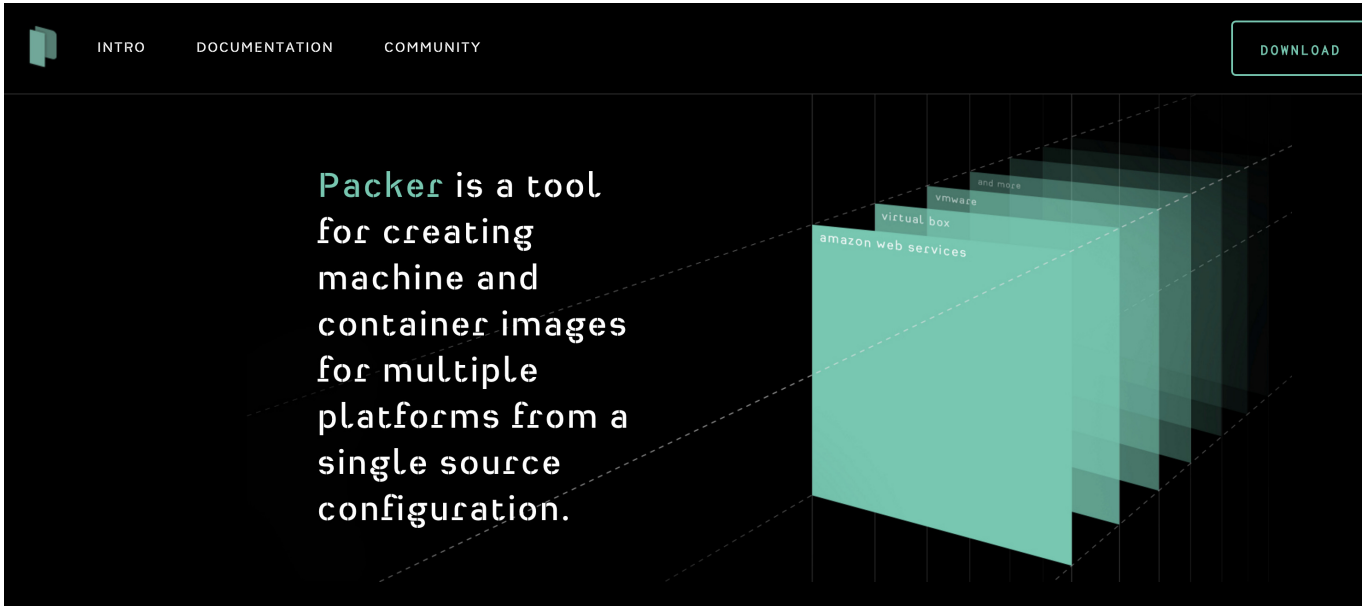
2 phase strategy

- 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

After packer age

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

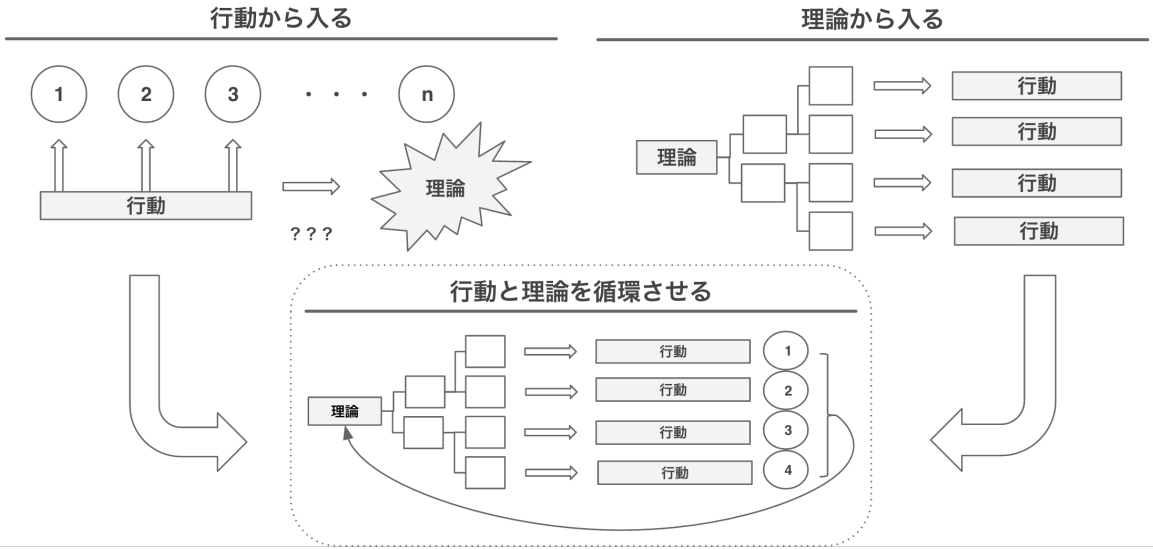


I think “Learning”

GMOペパボ株式会社

学習における2つのタイプ

行動することも理論を学ぶこともどちらも大切。フィードバックを循環させてできることを増やす





GMOペパボ株式会社

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 :)

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
```

www image

cloud-init

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

provisioner

```
puppet agent -t
set -e
monit stop unicorn
/usr/local/bin/globefish -w
rm -rf /var/www/deploys/minne/releases/*
rm -f /var/www/deploys/minne/current

# tar xf するだけで動くRails アプリケーションを取得

(snip)

# mackerel のホスト設定が packer 実行時のものとかぶらないように初期化

rm /var/lib/mackerel-agent/id
# cloud-init をもう一度動かすようにする準備

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
```

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

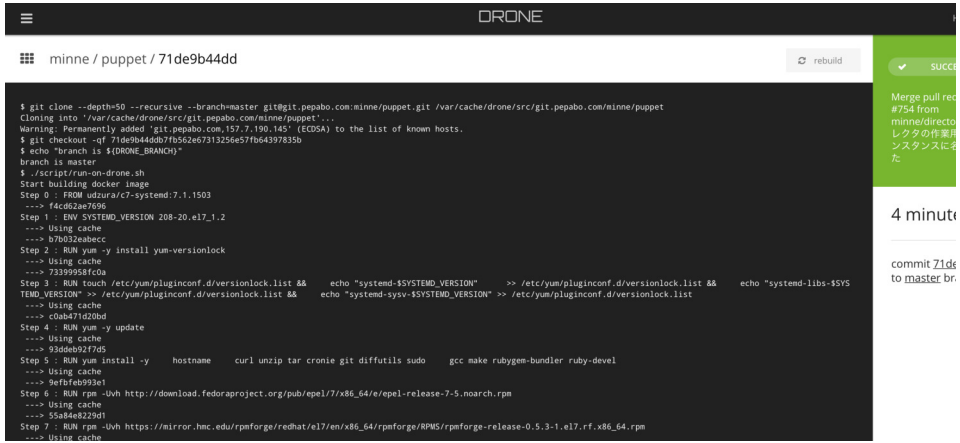
We can refactoring puppet manifests aggressively.

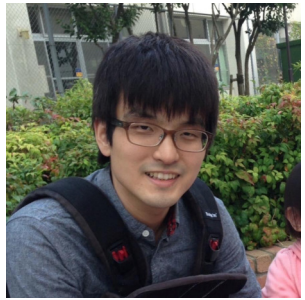
Drone CI

“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”





GMOペパボ株式会社

Serverspec

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

http://serverspec.org/

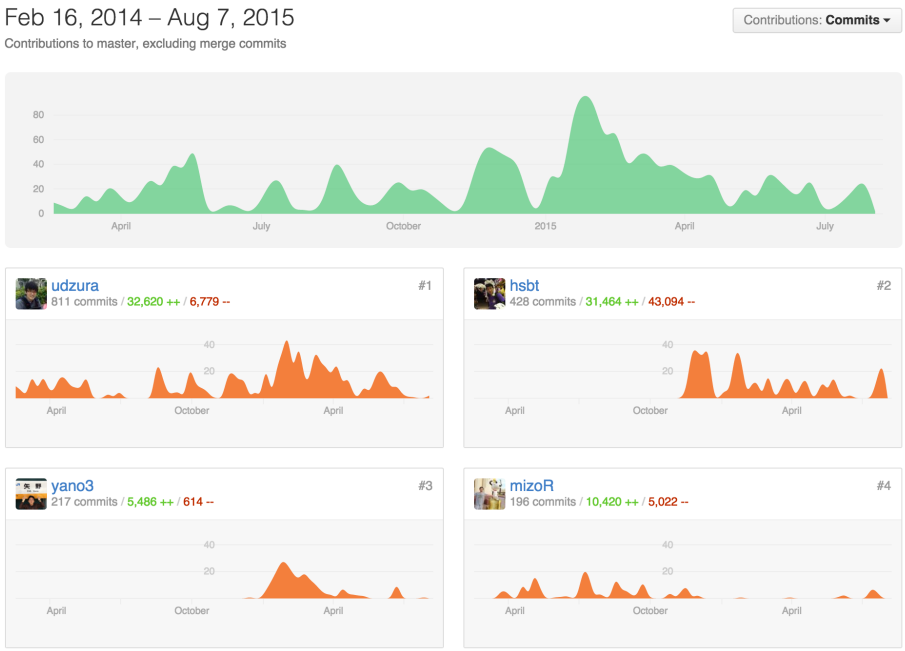
```
% rake -T
rake mtest              # Run mruby-mtest
rake spec               # Run serverspec code for all
rake spec:base          # Run serverspec code for base.minne.pbdev
rake spec:batch         # Run serverspec code for batch.minne.pbdev
rake spec:db:master     # Run serverspec code for master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



CentOS 7

Switch Scientific Linux 6 to CentOS 7

We can refactoring to puppet manifests with infra CI.

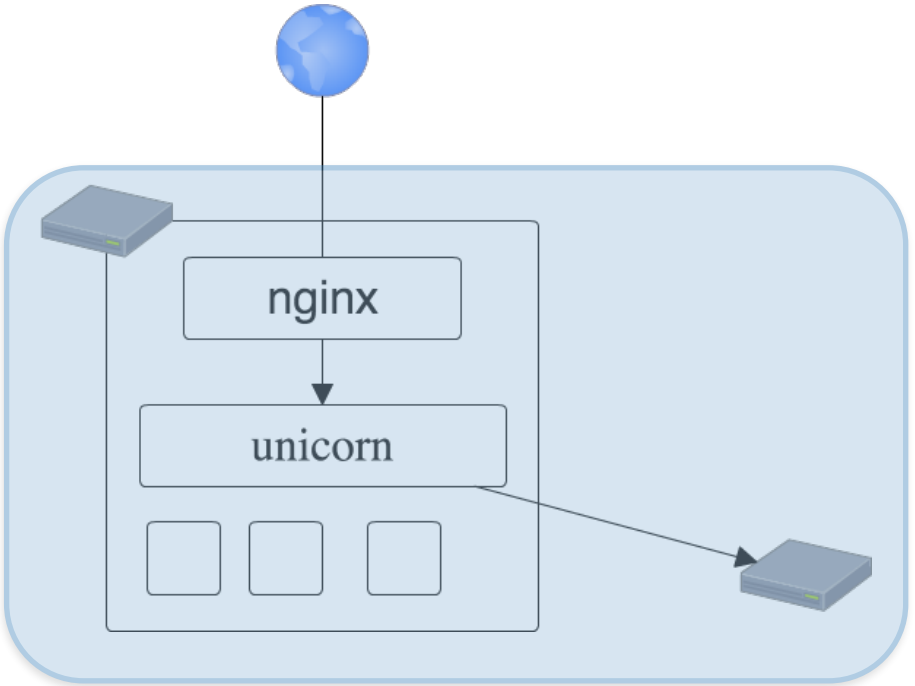
We added case-condition for SL6 and Centos7

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

How to test instance behavior

We need to guarantee http status from instance response.

We removed package version control from our concerns.



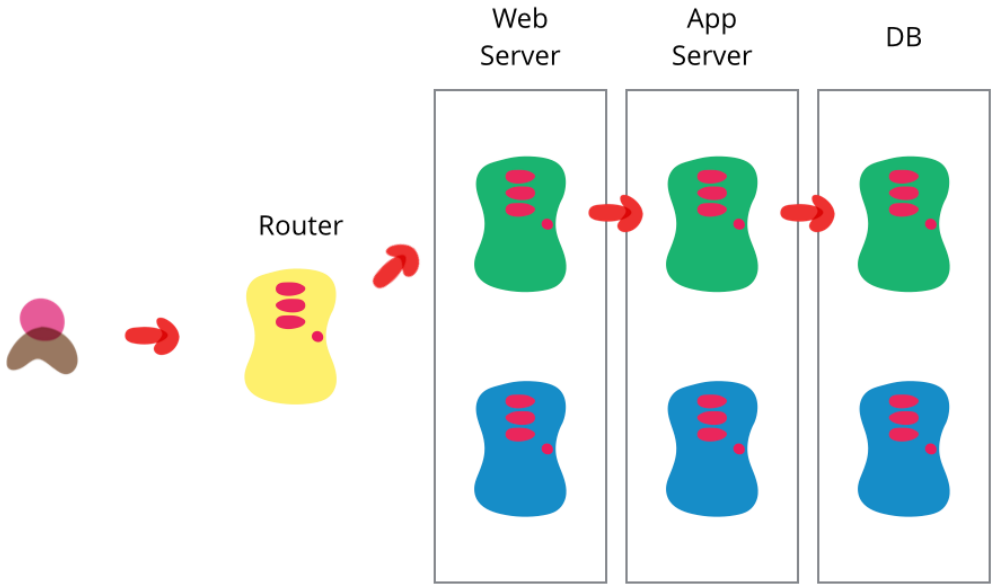
Check point 3

- Packer is best tool of Image creation
- Infra CI is over evaluation phase
- You can refactor provision manifests now

Bootstrap time = 3-5min

Blue-Green Deployment

What's Blue-Green Deployment



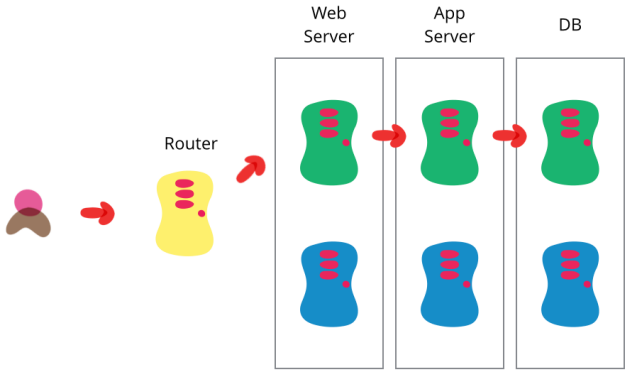
http://martinfowler.com/bliki/BlueGreenDeployment.html

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



Dynamic upstream with load balancer

ELB

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

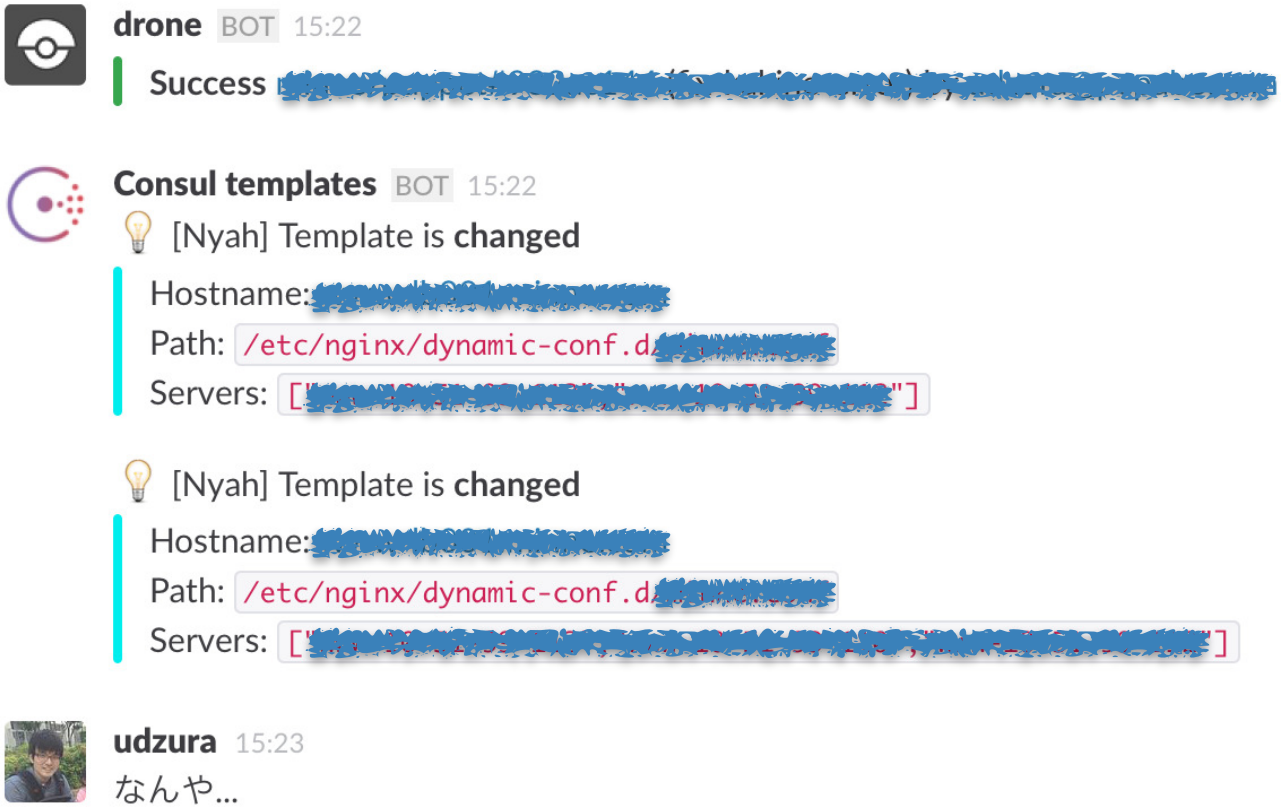
nginx + consul-template

- 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
```

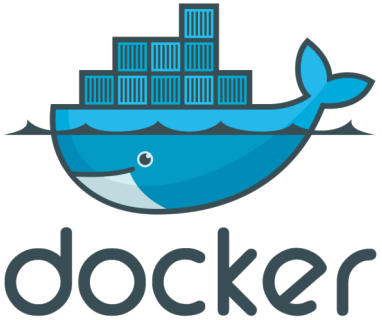
Check point 4

We can upgrade OS version in 3 minutes


＼ http://pepabo.com/recruit/ ／
採用
目的
採用
目的

Next step of our stage

- Automated all of test with image creation and launching
- Flexible architecture includes mutable roles
- Sync deployment with image creation cycle
- Use Docker



Enjoy Pythagoraswich Infrastracture



http://euphrates.jp/1859898