成長を加速する
minne の技術基盤戦略

変化するサービスとチームを支える

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”, …],
  sites: [“hsbt.org”, ruby-lang.org”, “rubyci.com”, “railsgirls.com”, “railsgirls.jp”],
}



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

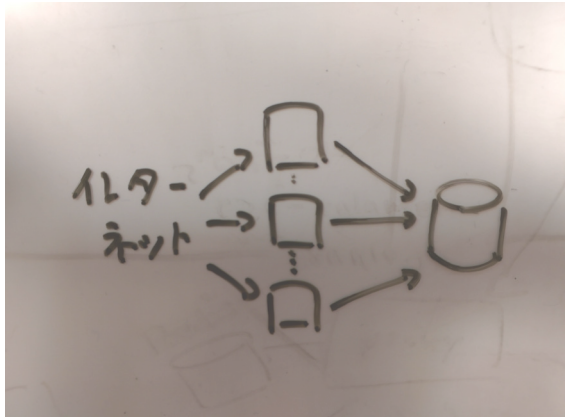
グローバルサイン認証サイト
SSL secured
クリックして確認
GlobalSign by GMO

株価情報

Do scale-out with automation

2014/11 の minne の状況

- IaaS の上で動くシンプルな Rails アプリケーション
- Rails が動いているサーバーは 6 台
- デプロイは capistrano 2 を使用
- ジョブワーカーは Web サーバーに同居
- 用途不明なサーバーもちらほら…



オペレーションは心温まる手作業

- 動いているサーバーを LB から外して “Golden Image” を作成し、インスタンスを複製
- 複製したインスタンスの設定変更は手順書をもとに手作業で実施
- おおよそ 4-6 時間の工程…

No ssh

“No SSH” ルールを作成し、Packer でイメージ構築を自動化



“No SSH” のコンセプト

ある程度の規模のサービスにとっては 1 サーバーは UNIX でいう 1 プロセスと同等である

プロセスに gdb でアタッチしたりしない
→ インスタンスに ssh ログインしない

プロセスのメモリを書き換えて変数を変えたりしない
→ インスタンスの設定ファイルを変えたりしない

No SSH によるサーバー構築の流れ

遅い処理

速い処理



Packer によるイメージ作成

- 公式 OS イメージ
  - プラットフォーム提供
- Minimal イメージ(phase 1)
  - Network, User, Package 設定のみ実行
  - puppet/chef, プラットフォームの cli ツールを追加
- Role 専用イメージ(phase 2)
  - 起動するだけで Role 特有のアプリケーションが起動

Minimal イメージ

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 イメージ

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/*
```

thor でオペレーションをコード化

packer の実行は ruby の thor から実行

```
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
```

イメージ作成

```
$ 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
```

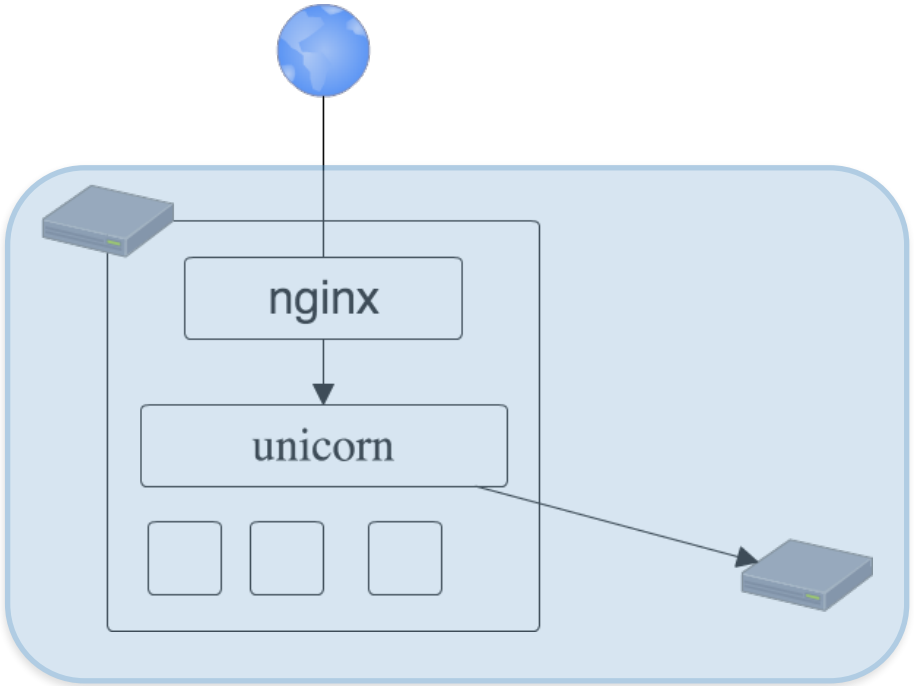
その他 IaaS 操作コマンド

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

インスタンス挙動のテスト

http(s) でアクセスした時の
挙動のみテストを行う

サーバー内部のパッケージ
個別のバージョンナンバー
などはテストの対象としない



インフラCIの導入

サーバーにインストール済みのパッケージや起動しているプロセスなどの詳細は Serverspec を用いて継続的にテストを実行

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

CIがあれば何でも(puppet マニフェストをアグレッシブにリファクタリング)できる!

Serverspec

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

http://serverspec.org/

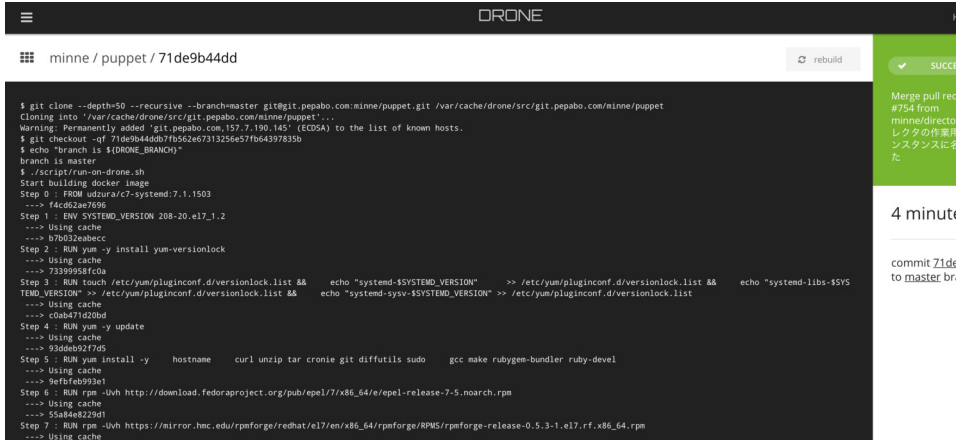
```
% rake -T
rake mtest              # Run mruby-mtest
rake spec               # Run serverspec code for all
rake spec:base          # Run serverspec code for base.minne.pbdev
rake spec:batch         # Run serverspec code for batch.minne.pbdev
rake spec:db:master     # Run serverspec code for master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

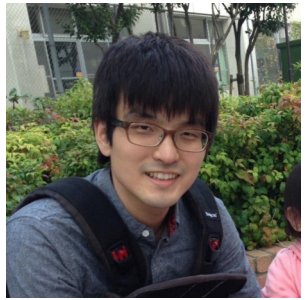
Drone CI

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

https://drone.io/

Drone CI は nyah と呼ばれる Openstack の上に構築





GMOペパボ株式会社

Integration tests with Packer

Packer の実行後にも Serverspec でテスト実行 (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 }}'"
  }
]
```

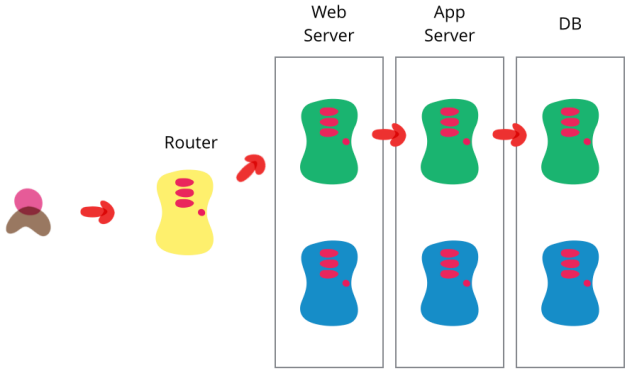
run-serverspec.sh

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

Blue-Green Deployment

Blue-Green デプロイの手順

1. 起動するインスタンスを Packer で作成し、hakata コマンドで起動
2. LB に接続して、“InService” となるまで待機
3. 古いインスタンスを廃棄



B-G デプロイの hakata コマンド

```
class deploy < Thor
    def blue_green
        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
    end
end
```

nginx + consul-template の様子



drone BOT 15:22

Success [redacted]



Consul templates BOT 15:22

[Nyah] Template is **changed**

Hostname: [redacted]

Path: `/etc/nginx/dynamic-conf.d[redacted]`

Servers: [redacted]

[Nyah] Template is **changed**

Hostname: [redacted]

Path: `/etc/nginx/dynamic-conf.d[redacted]`

Servers: [redacted]



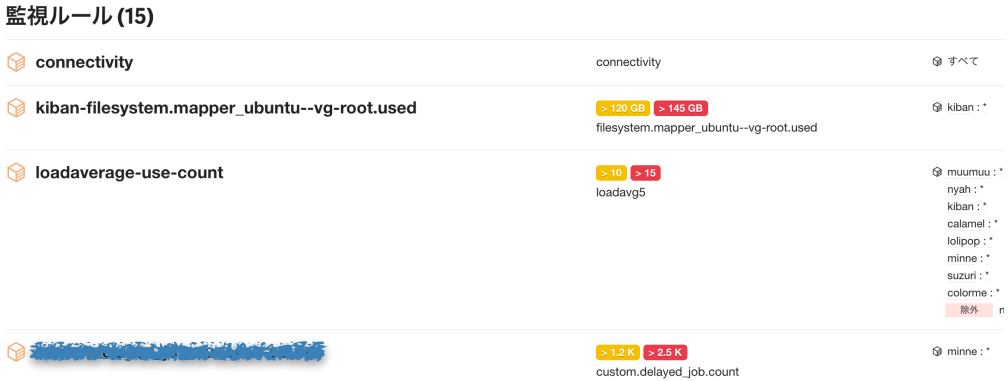
udzura 15:23

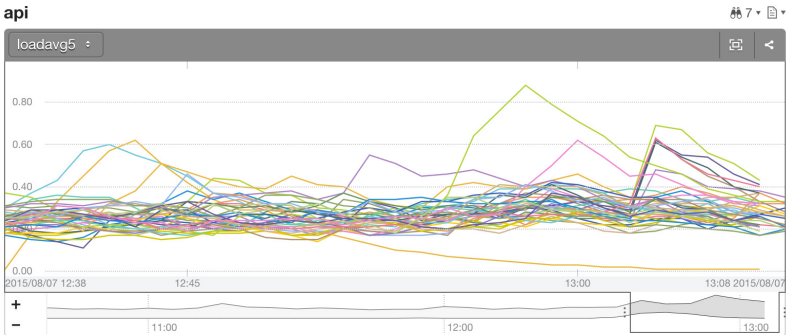
なんや...

Mackerel

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

https://mackerel.io



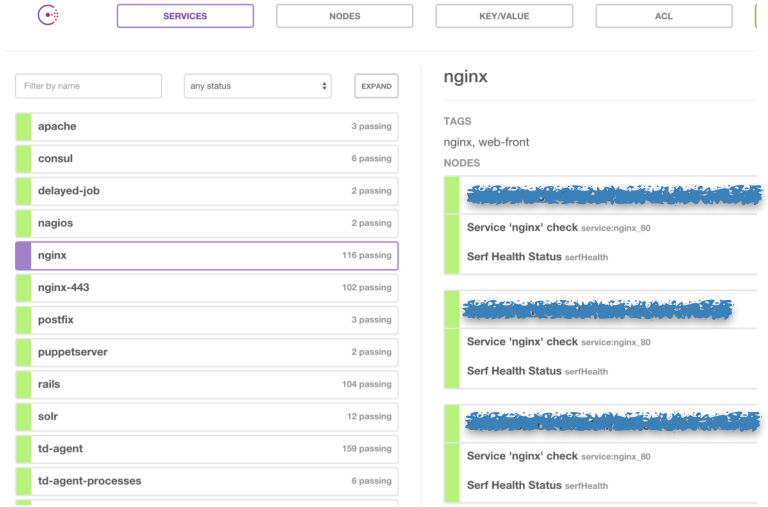


GMOペパボ株式会社

consul + consul-alerts

Disposable なインスタンスのプロセス監視は consul と consul-alerts で実行

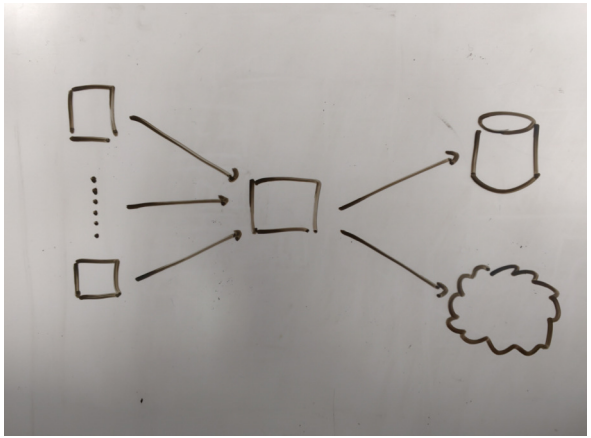
https://github.com/hashicorp/consul
https://github.com/AcalephStorage/consul-alerts





td-agent と log collector

動的に変化するインスタンスのログは td-agent で集約



```
<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>
```

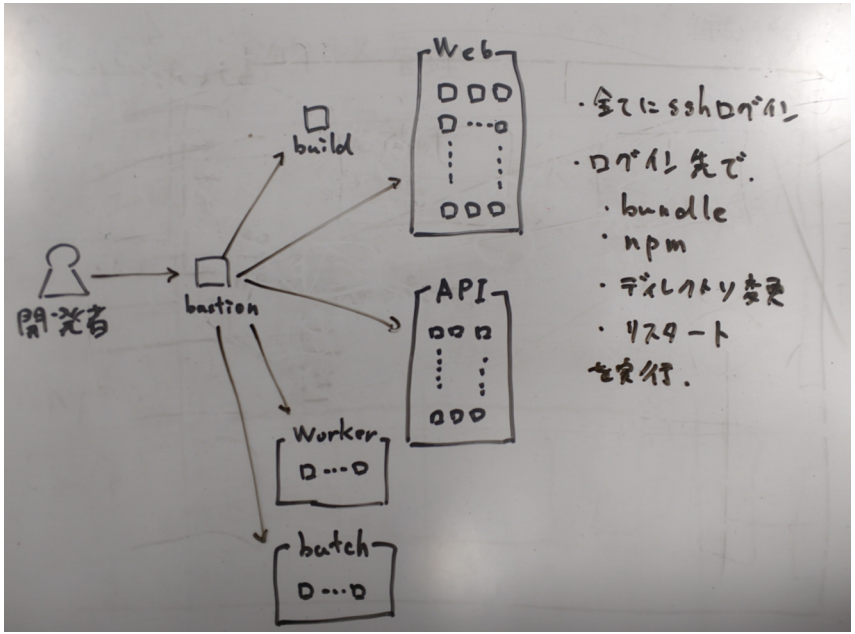
Large-scaled Deploy with Rails application

2015/11 現在の minne の状況

- Rails 4.2.4 and Ruby 2.2.3, MySQL 5.6.23
- Capistrano 3, stretcher + consul
- solr(with sunspot), delayed_job
- Models: 136, Controllers 143, Code to Test Ratio: 1:1.7
- サーバー台数 100台弱

capistrano によるデプロイ問題

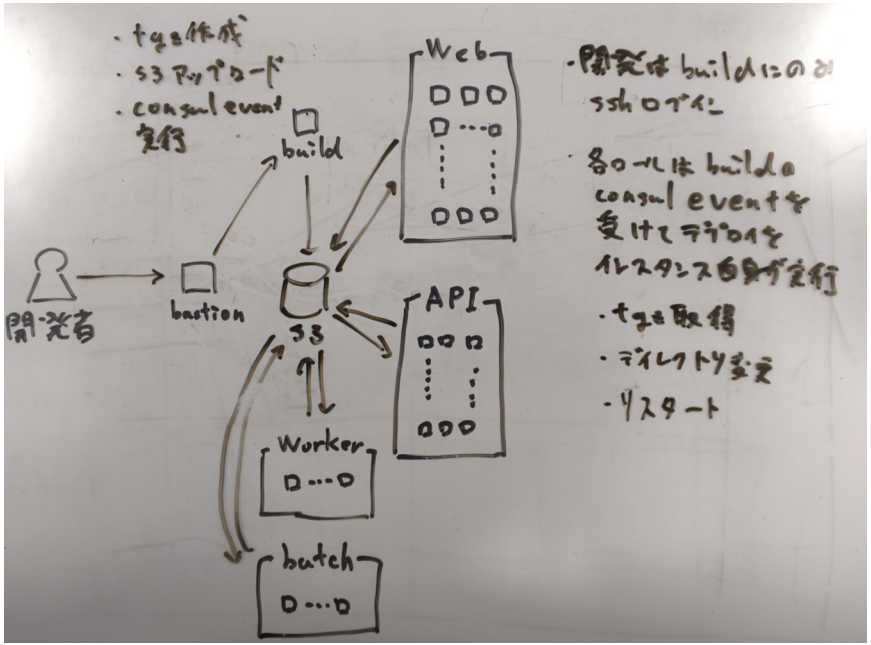
- ある日、社内の GHE が不定期に重くなるという報告
- GHE にログインしてプロセスリストを見てみると…
- git…git…git…(100個くらい)
- minne がデプロイすると全サーバーが GHE に git clone/fetch を実行する



consul with stretcher

https://github.com/fujiwara/stretcher

- トリガとして consul/serf のイベントを受け取って動作するプログラム
- インスタンスが自律的にコード更新を実行
- 更新するコードは s3 から取得



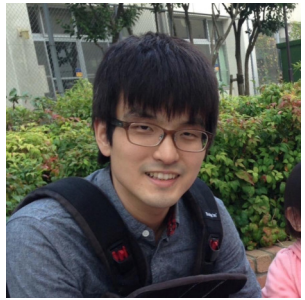
Bundled package of Rails application

Rails アプリケーションを Ruby だけあれば起動するような tgz を作成(bundle install/assets precompile 済み)

```
$ bundle exec cap production archive_project
```

capistrano を用いてビルド専用サーバーで各種タスクを実行後に s3 へアップロード

```
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]
```



consul watch と stretcher

consul の event を通知するために consul watch で stretcher を systemd でデーモン化

```
[Unit]
Description=Stretcher Deamon with Consul
Documentation=https://github.com/fujiwara/stretcher

[Service]
User=rails
Group=rails
EnvironmentFile=-/etc/sysconfig/consul
Environment="AWS_CONFIG_FILE=/home/rails/.aws/config"
ExecStart=/usr/bin/consul watch -type event -name <%= @event_name %> stretcher
ExecReload=/bin/kill -HUP $MAINPID
KillSignal=SIGINT

[Install]
WantedBy=multi-user.target
```

systemd への完全移行

supervisord や monit で動かしていた consul などを全て OS 標準の systemd へと移行

```
[Unit]
Description=Rack HTTP server for fast clients and Unix
Documentation=http://unicorn.bogomips.org/
[Service]
User=rails
Group=rails
EnvironmentFile=-/etc/sysconfig/unicorn
WorkingDirectory=/var/www/rails_applicaiton
PIDFile=/var/www/rails_application/shared/pids/unicorn.pid
ExecStart=/usr/local/rbenv/shims/bundle exec unicorn -c config/unicorn.conf -E <%= @environment %>
ExecReload=/bin/kill -USR2 $MAINPID
ExecStop=/bin/kill -QUIT $MAINPID
KillSignal=SIGINT
[Install]
WantedBy=multi-user.target
```

rails 向け stretcher manifests

yaml を erb から生成し、s3 にアップロードする cap task を作成

tgz 作成後に実行

```
src: s3://your-buckets-name/production/application-<%= env.now %>.tgz
checksum: <%= checksum %>
dest: /var/www/rails/releases/<%= env.now %>
commands:
  pre:
    -
  post:
    - ln -nfs /var/www/rails/releases/<%= env.now %> /var/www/rails/current
    - rm -rf /var/www/rails/current/log
    - ln -nfs /var/www/rails/shared/log /var/www/rails/current/log
    - mkdir -p /var/www/rails/current/tmp
    - ln -nfs /var/www/rails/shared/pids /var/www/rails/current/tmp/pids
    - ln -nfs /var/www/rails/shared/data /var/www/rails/current/data
  success:
    - <%= h[:cmd] %> && rm-releases /var/www/rails/releases 5
  failure:
    - cat >> /tmp/failure
    - (slack に失敗メッセージを通知)
- "*.pid"
- "*.socket"
```

capistrano との統合

```
namespace :minne do
  desc 'Deploy via Stretcher'
  task :deploy do
    set :deploying, true
    invoke "minne:archive_project"
    (ENV['ROLES'] || fetch(:minne_deploy_roles)).split(',').each do |target_role|
      on application_builder_roles do
        opts = ["-name deploy_#{target_role}_#{fetch(:stage)}"]
        opts << "-node #{ENV['HOSTS']}" if ENV['HOSTS']
        opts << “s3://your-buckets-name/manifest_#{target_role}.yml"
        execute :consul, :event, *opts
      end
    end
  end
  before 'minne:deploy', 'slack:deploy:starting'
  after 'minne:deploy', ‘slack:deploy:finished'
end
```

stretcher を用いて cap からデプロイ

```
$ bundle exec cap production minne:deploy
```

1. build サーバーで rails + bundler gems + node modules + assets 入りの tgz を作成、s3 にアップロード
2. stretcher 用の manifests を作成して s3 にアップロード
3. consul event を指定した roles に対して発行
4. event を受け取った role に所属するインスタンスで stretcher が起動、コードの更新

今後に向けて

今後に向けた技術基盤の刷新

検索によるユーザー価値の創造

- solr から elasticsearch への移行
- ハンドメイドならではの検索結果の表示

高速なサイトを目指して

- 高可用性ジョブキュー(sidekiq)への変更

効果的なモバイルUIの構築に向けて

- モバイルログ基盤の構築

今後に向けた技術基盤の刷新

IaaS のハイブリッド利用

- OpenStack への段階的な移行
- 10/21 時点で1-2割のサーバーが OpenStack で稼働中

完全オートスケールの導入

- OpenStack 上で実現するツールの開発

もっと
おもしろく
できる



10倍、100倍で ＼バーン！／

エンジニア大募集中！

ペパボをもっと技術的に＼バーン！／といかせるために、iOS/Android アプリエンジニア、Web アプリケーションエンジニア、インフラエンジニア、情報システムエンジニア、セキュリティエンジニア、技術基盤整備エンジニア、学生インターンなど、幅広い分野でエンジニアを積極的に募集しています。ペパボのエンジニアとランチをしながら楽しく会話できる「ペパランチョン」も定期的に開催しています。技術的探究心を持ち、チームメンバーとコミュニケーションをとりながらサービスを成長させていくことにコミットしたい！という方、エントリーをお待ちしています。

pepabo.com/recruit　pb_recruit　GMOペパボ株式会社

ペパボテックカンファレンス

第4回 EC編

PEPABO.TECH

これからのペパボの話をしよう





今年4月の開始以来、毎回ご盛況いただいております
ペパボテックカンファレンスですが、早くも第4回目を開催します。
今回は、有料ネットショップ運営サービス国内No.1
「カラーミーショップ」のエンジニアたちをフィーチャーする「EC編」です。

ペパボテックカンファレンスとは？

ペパボは、ロリポップ！や30days Albumなどのホスティング事業、カラーミーショップやminneなどのEC支援事業、JUGEMやブクログなどのコミュニティ事業といった領域において、多数のサービスを10年以上にわたって運営してきました。そして、その経験の中から、幅広い技術を蓄積してきました。

そんなペパボのエンジニアがお届けする技術トークは、ECサービスの根幹を成すショッピングカート機能をRailsとAngularJSで作りなおした話、流通額の飛躍的増加にそなえてインフラ基盤を次世代アーキテクチャに刷新した話など最新の技術に関する話もありつつ、より高いユーザ価値を提供するためのUX向上のための最新のアプローチや、日々のお問い合わせ対応を含めた運用にまつわる話までもりだくさん。ECサービス開発のすべてについて語り尽くします。

きっと面白い話が聴けるはずです。万障お繰り合わせの上、是非、ご参加ください。

なぜカラーミーショップ？

カラーミーショップは、今年でリリース10周年をむかえました。ネットショップ初心者をターゲットに「誰でも簡単にネットショップが作成できるサービス」として誕生して以来成長を続け、現在では時代の最先端をいく巨大ネットショップも多く抱えるサービスとなりました。多種多様なお客様すべてに満足していただくために、安定したサービス運営をしながらも、常に新しい技術を取り入れ改善に改善を重ねています。新しい技術のお話から、泥臭い日常の運用のお話まで、盛りだくさんでお話いたします。

対象者

- エンジニアとして成長していきたいけど、具体的に何をやればいいのかまだ見えていないという方
- 「ペパボのエンジニアって、実際どんなことやってるの？」というのを知りたい方
- どうやってエンジニアとしてこの先生きのこるかについて考えている方

イベント概要

日程：2015年11月12日（木）19時〜21時
参加費用：無料
席数：80



・イベント終了後に懇親会を予定しております。
・開催時間は変更する可能性がございます。
・詳しい内容と応募はこちらのページからご確認ください。

http://connpass.com/event/21552/