

成長し続けるインフラの 安定運用事情

Ryosuke Suto

2015/04/23

自己紹介

- 須藤 涼介 (すとう りょうすけ) @strsk
- 株式会社サイバーエージェント
- Amebaソーシャルゲーム専任のインフラエンジニア
- 4人で約30サービスを担当



アジェンダ

- ボーイフレンド（仮）のインフラ構成
- ソーシャルゲームのインフラに求められる要件
- 要件をクリアし、安定運用するために行っていること

ボーイフレンド (仮) IS 何？



声優総勢

30名以上

豪華声優多数参加！運命のカレに出会おう

ボーイフレンド（仮）とは？

- ・イケメンの声が聞ける学園恋愛ゲーム
- ・会員数200万人突破！
- ・豪華声優が30人以上参加！
- ・いろいろなイケメン（カード）と出会って
- ・好きなカレを自慢したり
- ・愛情を育んだりするゲームです

豪華声優多数参加！運命のカレに出会おう



ボーイフレンド（仮）の インフラ構成

声優総勢

30名以上

豪華声優多数参加！運命のカレに出会おう

Internet



プライベートクラウド

音声データを保存



Voice Server

NGINX

jetty://

Web Application Server

Cache Server



オンプレミス

mysql-master-ha(MHA)
で冗長化



Game Database



Game Event Database

スタンバイ状態の2台
が参照用スレーブ

豪華声優多数参加! 運命のカレに出会おう

ボーイフレンド（仮）構成

- 約70台
- プライベートクラウドとオンプレミスのハイブリッド
- 約半数がWebアプリケーションサーバ
- データベースはMHA for MySQLで冗長化
- 参照クエリはLBを経由しスレーブに
- PCIe型フラッシュストレージを搭載
- アクセスの多いイベント用テーブルを分割

なぜプライベートクラウド？

- 仮想化によって高性能サーバのリソースを効率よく利用できる
- 密度が高くなるためデータセンターも効率良く利用できる（電源注意）
- コストメリットがある
- 使いたいときにすぐ使える

なぜオンプレミス？

- I/Oがボトルネックになりがちな部分は仮想化のオーバーヘッドが大きくパフォーマンスが劣る
- PCIe型フラッシュストレージなど、ハードウェアに頼ってきた部分をカバーできない

ハイブリッドだけじゃない

- 某学園恋愛ゲームはすべてオンプレミスで150台弱
- 某仮想空間プリンセスバトルゲームはすべてAWSで150台オーバー
- すべてプライベートクラウドのサービスもあり
- 1000台前後？のインフラを運用



巨大なインフラ！

ソーシャルゲームに 求められるシステム要件


ソーシャルゲームの特徴

- 日々イベントを運用しているため負荷の状態も日々変化する
- サービス停止時間がそのまま機会損失につながる
- 行き廃りが激しい（厳しい世界）
- ネイティブ化、リッチ化によってリアルタイム性も重要になってきている

求められる要件

- キャパの把握、スケーラビリティは前提
- 日々の変化に対し**素早く**スケールできる
- 突発的な障害に対し**素早く**解決できる
- **素早く**立ち上げ**素早く**畳むことができる
- レスポンスも常に**素早く**

求められているのは速さ



巨大で動的なインフラ！

要件をクリアするためには

- やらなくて良い部分をやらない
- 手を動かさなくて良いところを自動化する
- 作業自体を効率化する
- 作業の再現性を高くする

要件をクリアする
= 安定運用につながる

安定運用するために
してきたこと

サーバ構築時

サーバ構築時に起きた問題

- 構築作業に時間がかかる
- 新規で構築したマシンと既にサービスインしているマシンで設定が違う
- 新規構築が前任者だったため手順がわからない（探しても見つからない）

事案①

「今度CM打つことになったのでサーバ増強
お願いします！」

「了解です！ちなみにいつからですかね？」

「週明けからの予定ですー」

「わ、わかりました（震え声）」



プロビジョニングツールの導入

- Infrastructure as Code
- [Chef](#), [Ansible](#)を利用
- サーバの状態をコード化してGithubで管理
- コマンド一発で何回目の実行でも同じ状態に収束するという観点（冪等性）
- コード化されているので誰が実行しても同じ

Chef

```
package "httpd" do
  action :install
end

service "httpd" do
  supports :status => true, :restart => true, :reload => true
  action [ :enable, :start ]
end
```

- サーバの状態を記述
- httpdのインストール、自動起動設定、起動

Ansible

```
- hosts: localhost
  sudo: yes
  tasks:
    - name: be sure httpd is installed
      yum: name=httpd state=installed

    - name: be sure httpd is running and enabled
      service: name=httpd state=running enabled=yes
```

- 同じ処理をAnsibleで書いた場合
- httpdのインストール、自動起動設定、起動

導入理由

- Chefは識者がいたためm(_ _)m
- Chefは全サービス共通で使えるように管理されていて便利な分、複雑なので身軽に使えるツールが欲しかった
- Ansibleはクライアントにインストールが必要なく、学習コストもchefほどではなかったため

導入してみた結果

- pros

- 構築が圧倒的に楽になった
- 設定が違ふといった事案が減った
- コードを共有して学べる文化が生まれた

- cons

- プロビジョニングツール自体の仕様にハマることがある . . .

サーバ構築以外でも

- [Terraform](#), [Packer](#), [Roadworker](#)
- 主にAWSでの利用
- AWSのネットワークや初期構築をコード化
- 全てのサーバで共通の初期設定を行ったAMIを作成
- DNSレコードをコード化して管理
- 詳しくはWebで

監視設定時

監視設定時に起きた問題

- 増設したサーバ群の設定を追加したけどIPがカブって漏れが発生していた
- 増設したマシンを監視に追加したのに監視すべき項目が監視されていなかった

事案②

「監視設定をお願いしますー」

「了解です！」

～時は流れ～

「あれ、こないだ入れたマシンだけリソース見れないですね」

「か、確認します（震え声）」



ZABBIX



監視設定～開始を自動化

- 監視サーバでの手動設定をやめる
- 監視対象のセットアップ完了時に監視を開始する
- [Zabbix](#), [Sensu](#), [Mackerel](#)を利用
- プロビジョニングツールで構築した時点で監視対象になる

Zabbix

ZABBIX Help | Get support | Print | Profile | Logout

Monitoring | Inventory | Reports | Configuration | Administration

Dashboard | Overview | Web | Latest data | Triggers | Events | Graphs | Screens | Maps | Discovery | IT services

History: トリガーの設定 >> ダッシュボード >> ユーザープロフィール >> Dashboard >> Overview

PERSONAL DASHBOARD

Favorite graphs

- vSphere_001: CPU utilization
- vSphere_002: CPU utilization
- vSphere_003: CPU utilization

Favorite screens

- Zabbix server performance
- JBoss performance
- Oracle RAC
- Network map

Favorite maps

- Network devices
- VMWare production

Status of Zabbix

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (monitored/not monitored/templates)	85	47 / 0 / 38
Number of items (monitored/disabled/not supported)	502	493 / 0 / 9
Number of triggers (enabled/disabled) [problem/ok]	291	291 / 0 [10 / 281]
Number of users (online)	2	1
Required server performance, new values per second	7.7	-

Updated: 02:41:40 AM

System status

Host group	Disaster	High	Average	Warning	Information	Not classified
Business System	0	0	0	0	0	0
Clouds	0	0	0	0	0	0
Database servers	0	0	0	0	0	0
JBoss instances	0	0	0	3	0	0
Network Devices	0	0	0	0	0	0
Private Cloud	0	0	0	5	0	0
Web servers	0	0	0	0	0	0
Zabbix servers	0	0	0	2	0	0

Updated: 02:41:41 AM

Host status

Host group	Without problems	With problems	Total
Business System	17	0	17
Clouds	2	0	2

Zabbix

- Zabbixエージェントを監視対象にインストールしてホストとして登録するとZabbixサーバーがホストと通信し監視を行う
- 監視の構造が慣れないとわかりにくい
- メトリクス取得も死活監視もだいたいなんでもできる
- UIがわかりづらい

Zabbixで自動化

- Zabbix APIを利用してホストを追加
- プロビジョニングツールと連携し、実行時に自分をホスト追加するためのリクエストを送信
- 各ミドルウェアの監視項目（テンプレート）もミドルウェアのレシピに紐づく

<http://www.zabbix.com/jp/img/zabconf2013/presentations/12-cyberagent.pdf>

Sensu

Browser address bar: /stashes

Navigation menu: Sensu-Admin | Events | Clients | Stashes | Checks | Downtimes | Aggregates | Logs | Stats | Account | Users | Settings | Logout

[Create Stash](#) [Delete All Stashes ▾](#)

Stashes (154)

Key	Description	Owner	Expires in	Set	Action
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete
silence/	No reason given	<input type="text"/>	Never	9days ago	Delete
silence/	RELEASE: 2014-05-07 14:47:47.257030	jenkins	Never	<1min ago	Delete

Sensu

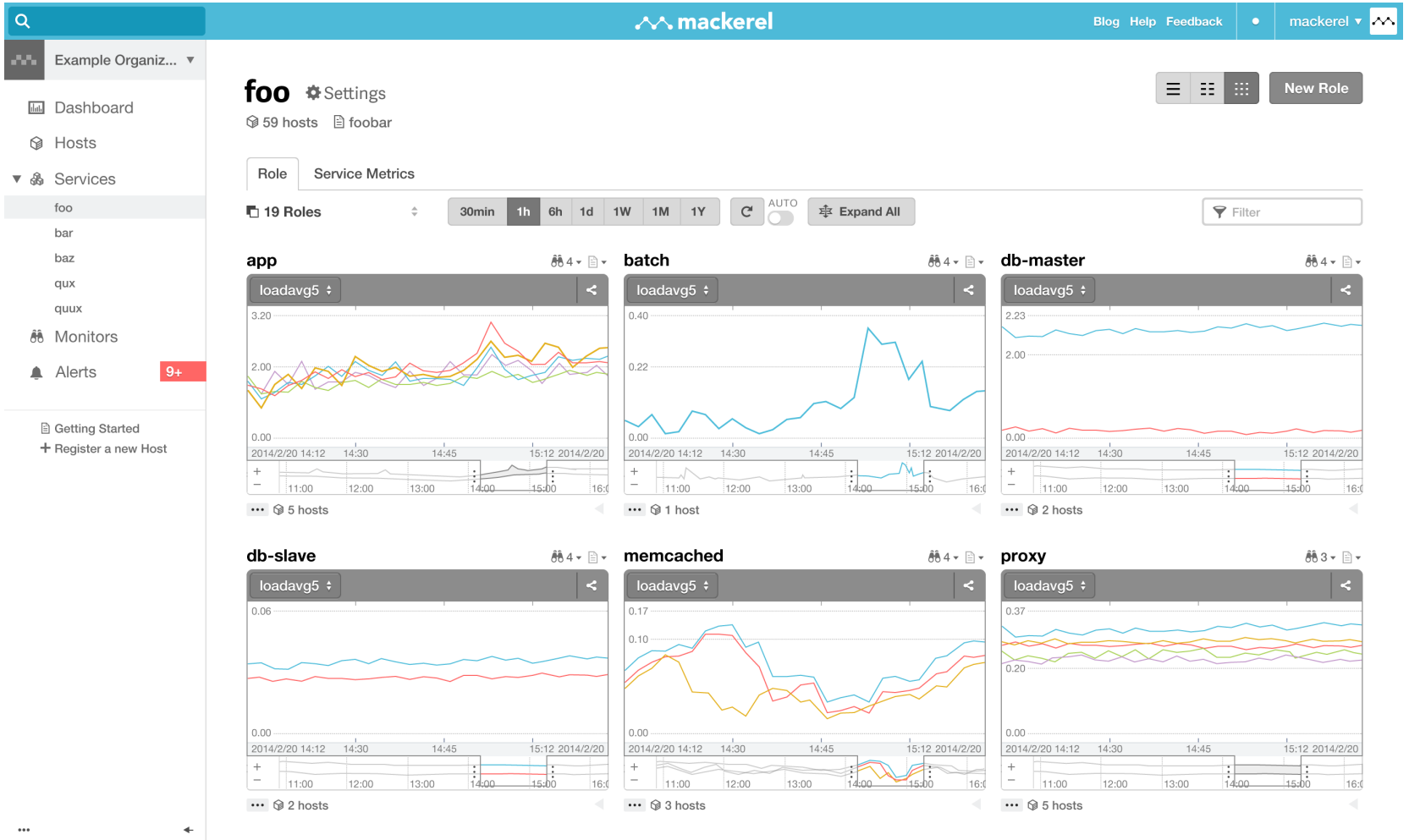
- Sensu Clientを監視対象にインストールして設定をすると自動的にSense Serverに認識される
- Sensu ServerとSensu Clientの通信はRabbitMQを介して行われる
- WebUIのツールは複数あって好みが変わる

Sensuで自動化

- Sense Clientの設定に利用するRabbitMQのIPを指定して起動する
- プロビジョニングツールと連携して、実行時に起動

<https://hiroakis.com/blog/2014/05/08/%E7%9B%A3%E8%A6%96%E3%82%B7%E3%82%B9%E3%83%86%E3%83%A0%E3%82%92sensu%E3%81%AB%E5%88%B7%E6%96%B0%E3%81%97%E3%81%9F/>

Mackerel



Mackerelの場合

- 監視型SaaS
- 有料
- 監視対象をRoleで管理し、監視を行う
- メトリクスもRoleごとに見ることができる
- SaaSなので監視サーバを立てる必要がない

Mackerelで自動化

- 登録したアカウントで取得したトークンをクライアントの設定に指定して起動
- プロビジョニングツールと連携して、実行時に起動

導入理由

- Zabbix,Sensuは識者がいたためm(_ _)m
- Mackerelは監視サーバの運用コストを省
力化したかったなので検証導入中。メトリ
クスも見やすい。

導入してみた結果

- ・ 監視漏れがなくなった
- ・ 設定漏れもなくなった
- ・ いいことしかない

アラート発生時

アラート発生時に起きた問題

- アラートが鳴ったけど、サービスに影響あるのかわからない
- サービス影響のないアラートが多すぎてよくわからない



事案③

「なんかアラート来ましたね」

「ゲームは普通にプレイできてます」

「重いとかも特にはないかなー」

「と、特に問題なさそうですね…（震え声）」

アラートの精査

- ひとつひとつ整理する
- しきい値変えたほうが良いものは変える
- 出さなくて問題なさそうなら出さない
- アラートのレベルを適切に調整する
- 必要な監視項目を精査する
- 地道な活動が安定化につながる

属人性問題

属人性問題

- ・ 見るシステム数が多いため全員が全システム、リスクを把握するのが困難
- ・ 人と仕事をしている以上、偏りを失くすことは厳しい
- ・ とはいえできることはある

事案④

「おつかれさまですー」

「おつかれさまですー」

「先日のDB負荷の件で何かわかりました？」

「…あーはいはい、あの件ですねー（棒）」

属人性を減らすために

- 作業のコード化、自動化
- [Trello](#)を使ったタスク共有の仕組み
- 朝会でのアラートチェック、リスク共有
- すぐチャット、すぐ話す
- [Confluence](#) (社内Wiki) にとにかく書く
- 重要なタスクはプロジェクトにして全員でやる

やってみた結果

- 誰かが手を離せないときに他の人が対応できる（できてないところもある）
- 誰かが急に休んでもそこまで困らない
- メンバー全員が全体のタスクと重要度を把握できるようになってきている
- しかしまだまだできてないことも多い

現在とこれからの課題

現在

- 安定運用はできてきている
- インフラが起因するサービス停止はほぼなくなった（たまに…）
- 夜中に起こされることもほぼない(´ω`)スヤ

これからの課題

- 負荷やレイテンシとの戦いは続いている
- AWSのコスト意識を強くしたい
 - オートスケーリングができていない
- まだまだスピードが足りない
- スキル、ナレッジの偏りもまだまだ改善中

まとめ

- 開発スピードを上げれば安定運用につながるし、安定運用しようと思ったら開発スピードが上がるとも言える
- 規模が大きくなったときにやり方を変えるのではなく、規模が大きくなったときにスケールできるやり方を常に考えたい
- 問題意識を持って、地道にできることを増やしていく

ありがとうございました