CIサーバを制圧せよ！

プロジェクトメトリクスと自動化技術の活用よる混乱の収拾と「最強」の組織の育成

Apr/16/2015

Hiroyuki Ito, Kazuhisa Naoi
Rakuten, Inc.
http://www.rakuten.co.jp/

楽天 Rakuten

自己紹介 (1)

伊藤 宏幸 (The Hiro)

[@hageyahhoo]

- アジャイルコーチ
- TDDグループ（当時）

自己紹介 (2)

直井 和久

楽天株式会社

トラベルサービス開発・運用部

トラベルサービス開発課

Travel Extranetグループ

マネージャー

この物語は、ここでの改善に挑んだ、実践者達の記録である。

お伝えしたいこと

1. プロジェクトメトリクス

2. 戦略策定の重要性

3. Infrastructure as Codeの有益性

4. Technology-Driven Development

特にお伝えしたいこと！！

1. プロジェクトメトリクス

アジェンダ

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー・チームを育成していくか？

5. 結論

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー・チームを育成していくか？

5. 結論

2014年6月末

Technology-Driven Development: Using Automation and Development Techniques to Grow an Agile Culture

HIROYUKI ITO, DEVELOPMENT PROCESS OPTIMIZATION DEPARTMENT, RAKUTEN, JAPAN

In this experience report, I present a new mechanism called "Technology-Driven Development". "Technology-Driven Development" stands for three purposes: The first one is to make the work efficient. The second one is to develop cooperative relationships between the team members and stakeholders. The third one is to drive learning of the team members by technical practices and methods such as Continuous Integration [5] / Continuous Delivery [8] (hereinafter called the "CI/CD"), TDD (Test-Driven Development) and BDD (Behavior-Driven Development).

The "Technology-Driven Development" mechanism has been chosen not only as a technical foundation for developing new smartphone application, but also as a driver for the team that consisted of young and immature members to learn new skills. This way the team members learned the skill for developing software and solving problems comparably or even better than the senior members. Moreover, this mechanism grew the voluntary and supportive culture in our team.

1. INTRODUCTION

Over the years, the main purpose of automation techniques has been considered as a way of making the work efficient. Certainly automation can reduce manual operations, operational errors, and work hours. Originally we used it to work more efficiently. Although making the work efficient is valuable, there is more to software product development, namely learning and collaboration. Learning is necessary to create the software right. Collaboration is the key factor to create the right software with the team members and stakeholders. At the end of April 2013, I started supporting one new project as an "Agile Coach", a dedicated role in our company to educate the team members on agile practices, techniqu
Through this project, I discovered addition
techniques that drive learning and acceler
established as a new model for creating an agil

In this paper, first I explain our challeng
the concrete mechanism of "Technology-Driven
of making the work efficient, of learning, an
possibilities and the future are discussed.

Agile2014へ提出する論文と
まさに格闘中の時期…

トラブル、襲来

CIサーバの負荷が高すぎて、
期限通りに**プロダクトを**
リリースできない！

これ以上CIサーバの
ジョブを増やしたら、
サービスを提供できなくなる！

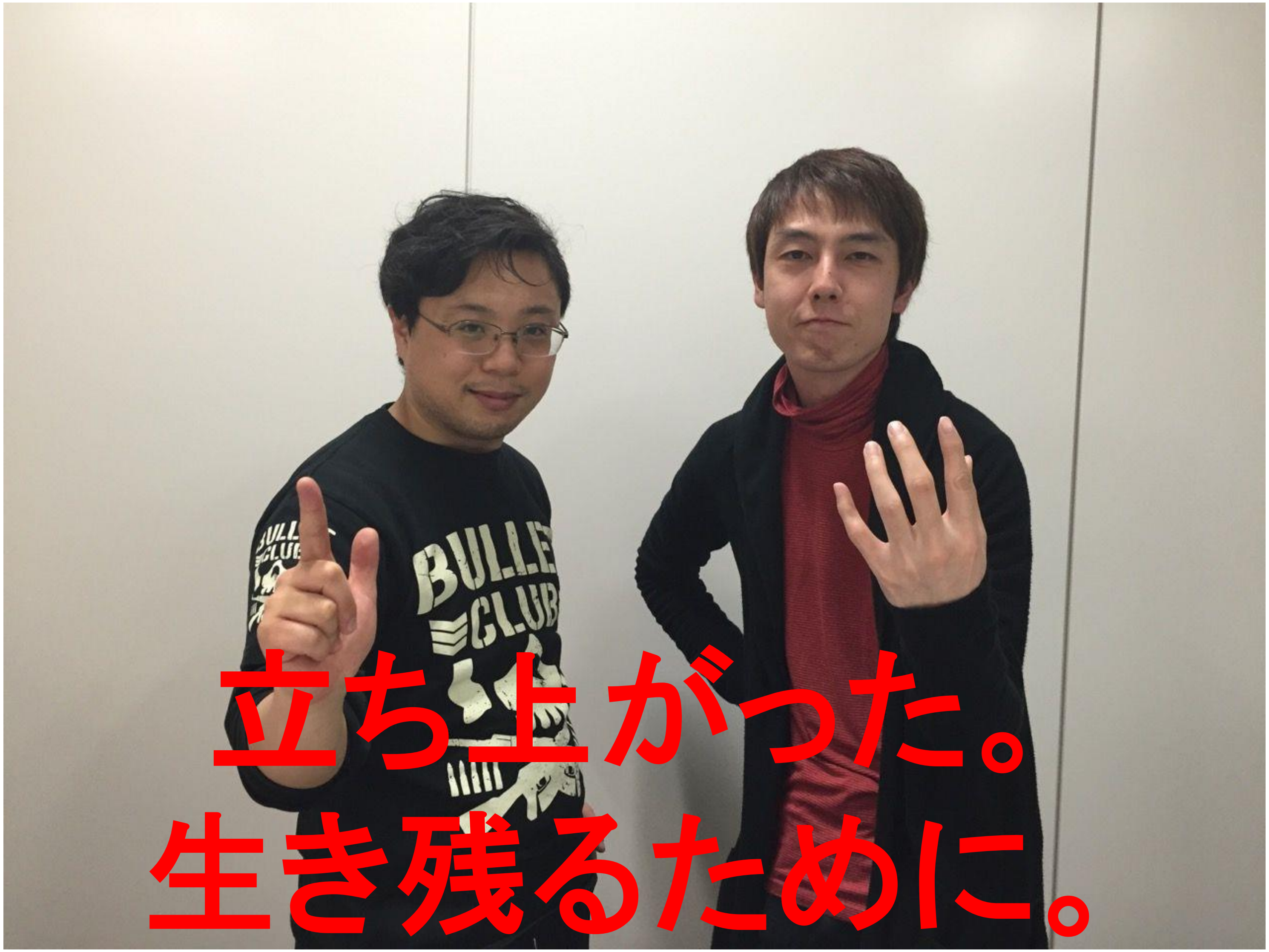
当時の我々の状況

<u>とあるQAプロジェクトの自動化を主導中</u>

- ビルドパイプラインの構築
- リリース自動化の実現
- スモークテストの構築

<u>メインシステムの多言語化対応をマネージャとして主導中</u>

お互い、CIサーバの設定・監視・運用には責務を持っていなかった。


立ち上がった。
生き残るために。

原因分析結果

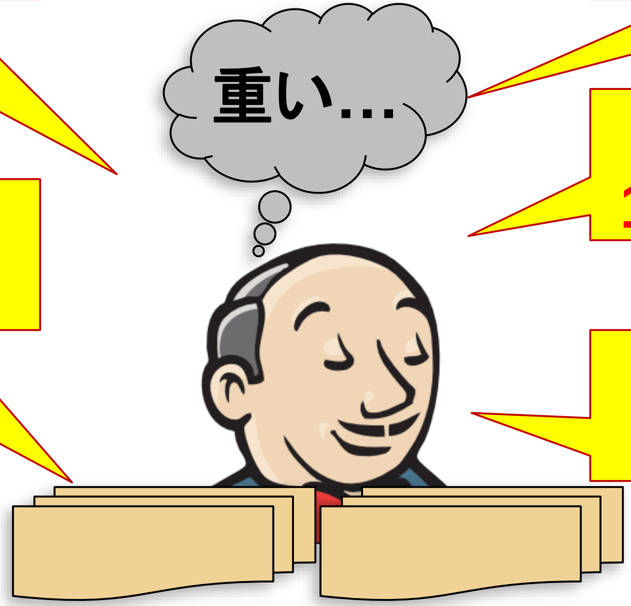
スタンドアローン構成

誰も監視･メンテしていない！

Load averageが10を超えるのはザラ

400以上のジョブ（さらに増加の傾向）

実行待ちキューが10前後なのもザラ



そもそも**画面すら開けない**ことが多かった。

- 「Script Console」を試すこと自体がバクチ☹

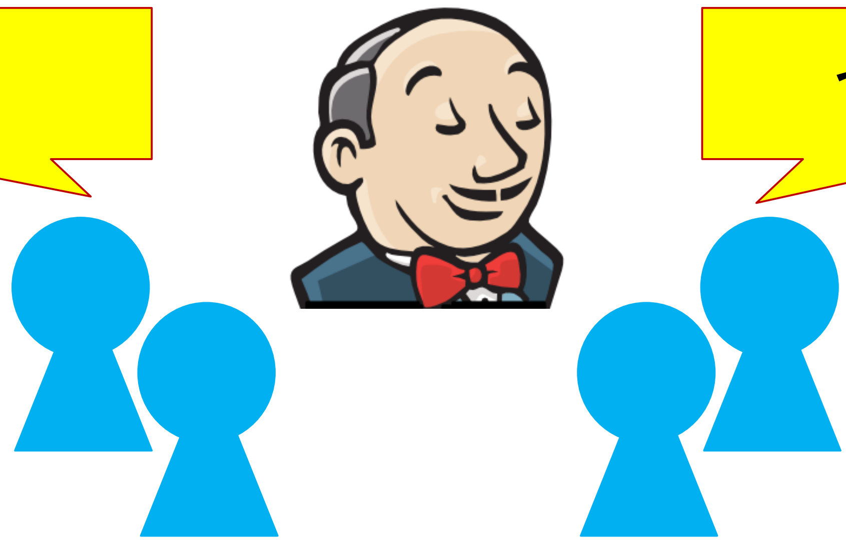
真因分析結果

組織横断的に、CIサーバ専任で動けるチームがない！

やるぞ～！

つくるぞ～！

有志の
エンジニアたち



元々、**ボランティア**で構築・運用されていた。

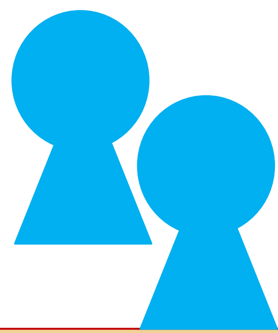
真因分析結果

組織横断的に、CIサーバ専任で動けるチームがない！

仕事多い (><)

誰かがきっと
何とかしてくれる…！

有志の
エンジニア
たち



- 問題が起きても**放置**されるようになってきた。
- サーバを**常時監視する仕組みがなかった**。

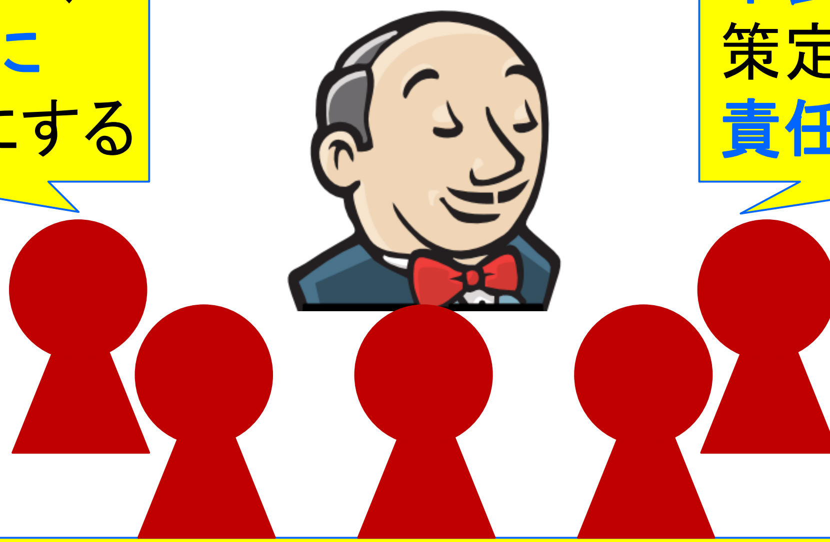
最初に実施した施策

CIサーバ専任の組織横断的チームの設置を提案した

負荷増大に対して、
長期的・根本的に
対応できるようにする

中長期ビジョンの
策定・実現にも
責任を持つ

常設
チーム



正常稼動しないことが**業務損失**になる現状、
常設・専任チームによる解決が必須と判断。

Jenkins Consolidation Team、結成。

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー・チームを育成していくか？

5. 結論

新たな課題

何を成果として見せれば良いのか？

自分たちはどこに向かえば良いのか？

自分たちは正しい方向に進んでいると
言えるのだろうか？

楽天 Rakuten

施策初期の悩み

仕事である以上、**成果を**マネージャ層・
ステークホルダーに**「みせて」「喜んでもらう」**
必要がある

でも、そもそも**CIサーバの成果**って、
何をどう見せれば良いのだろう？？？

試してみないと分からない仕事のため、
常に振り返りながら新しい施策を打ち立て
実施していく必要があった

解決策としてのプロジェクトメトリクス

成果としてマネージャ層・ステークホルダーに
「みせ易く」「うれしい」数値を見つける

変化の見える数値を探す

定期的に数値をトラックし、
特に施策実施前後の変化に注目する

数値自体の有効性を常にチェックし、
数値の変化の内容次第では、施策を見直す

プロジェクトメトリクスの前提として
基本戦略・ロードマップを
まず最初に明確化した。

特に注意した点

プロダクトに関わる**（極力）全ての人が合意できる目標**になっているか？
（例）インフラ担当者もOKできる内容か？

エンジニア視点だけで満足した目標になっていないか？
- ビジネス価値の観点も含むべき

持続可能な施策となっているか？
- ハネムーンナンバーは高くないか？
- 施策をリードできる人材は十分か？

基本戦略

1. 取り急ぎCIサーバの負荷を減らす

- 高負荷ジョブを一時停止するなどして、一時しのぎと改善の時間稼ぎをする。
- 負荷分散用サーバを調達・整備し、ジョブを各サーバに「一時疎開」させる。

2. CIサーバのボトルネックを解消する

- CIサーバ群をマスタースレーブ構成に改め、負荷分散を容易化する。
- 既知の技術負債等を1つ1つ解決し、CIサーバ群を「クリーン」にしていく。

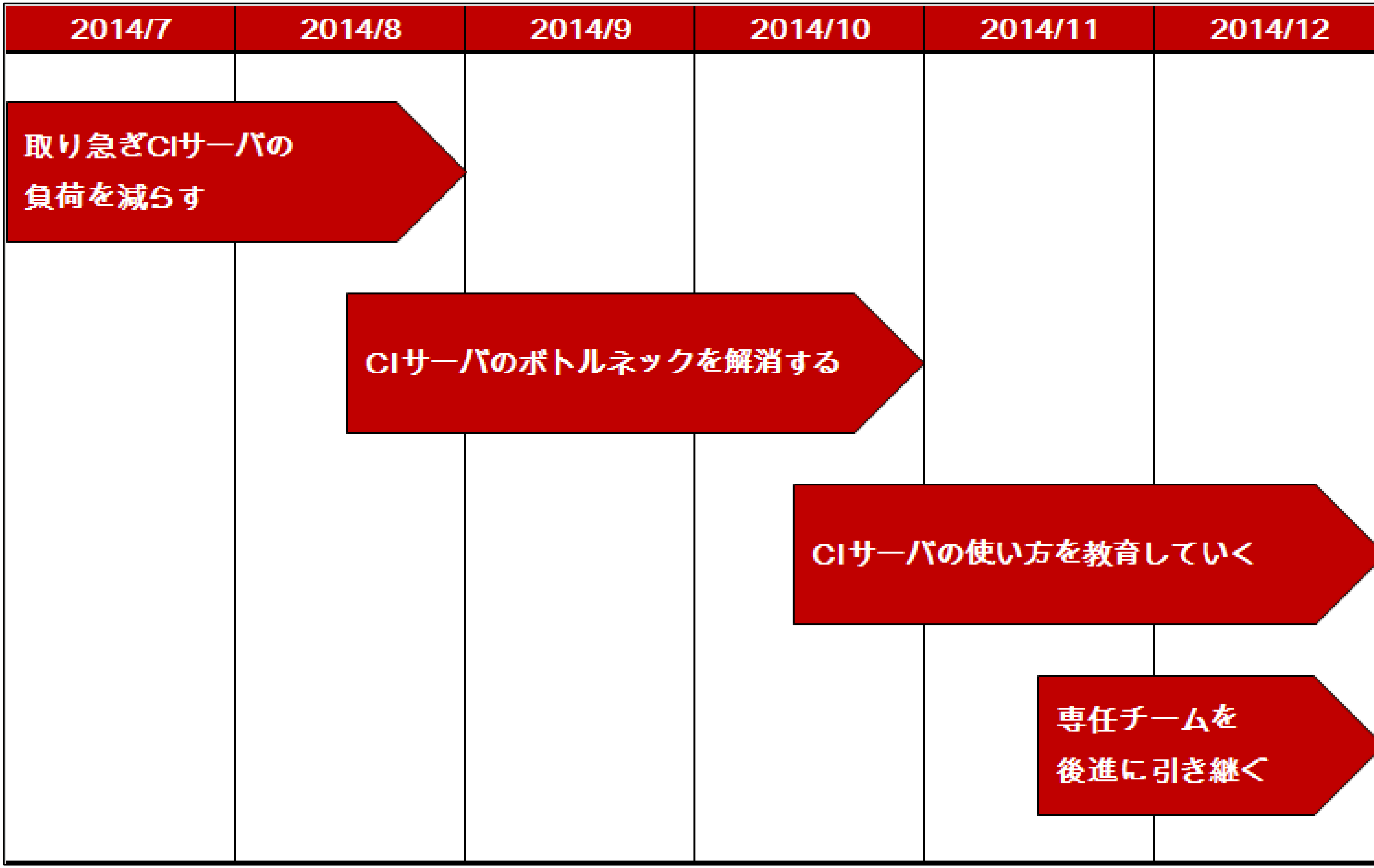
3. CIサーバの使い方を教育していく

利用者（エンジニア）にCIサーバの「正しい」使い方を教えることで、
技術負債・トラブルの芽を摘むと同時に、利用者の育成も図る。

4. 専任チームを後進に引き継ぐ

- 伊藤・直井がリードする体制から、育成も兼ねて後進にタスクを引き継ぐ。
- 結果として、CIサーバの「正しい」使い方をさらに広めることも図る。

ロードマップ


2014/7
2014/8
2014/9
2014/10
2014/11
2014/12
取り急ぎCIサーバの負荷を減らす
CIサーバのボトルネックを解消する
CIサーバの使い方を教育していく
専任チームを後進に引き継ぐ

苦悩と歓喜の始まり

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー･チームを育成していくか？

5. 結論

ご覧の有様

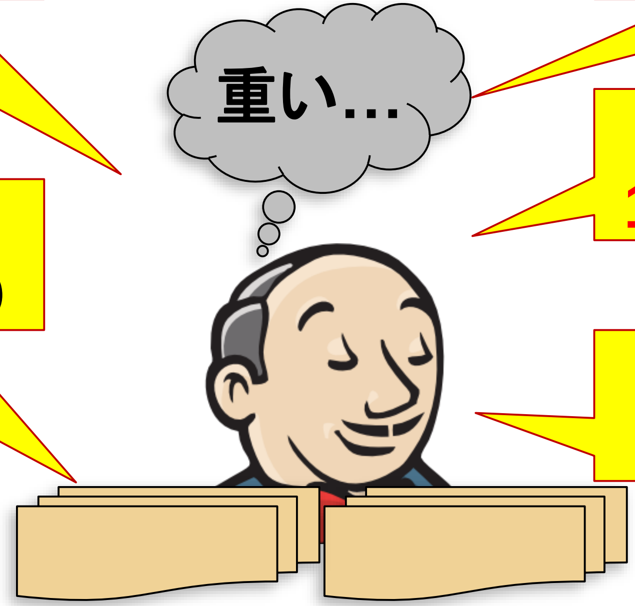
スタンドアローン構成

誰も監視・メンテしていない！

400以上のジョブ（さらに増加の傾向）

Load averageが10を超えるのはザラ

実行待ちキューが10前後なのもザラ



まずはこれを解決する！

段階的なアプローチを採用

その時点で計測可能なプロジェクトメトリクスに絞って見せることとした

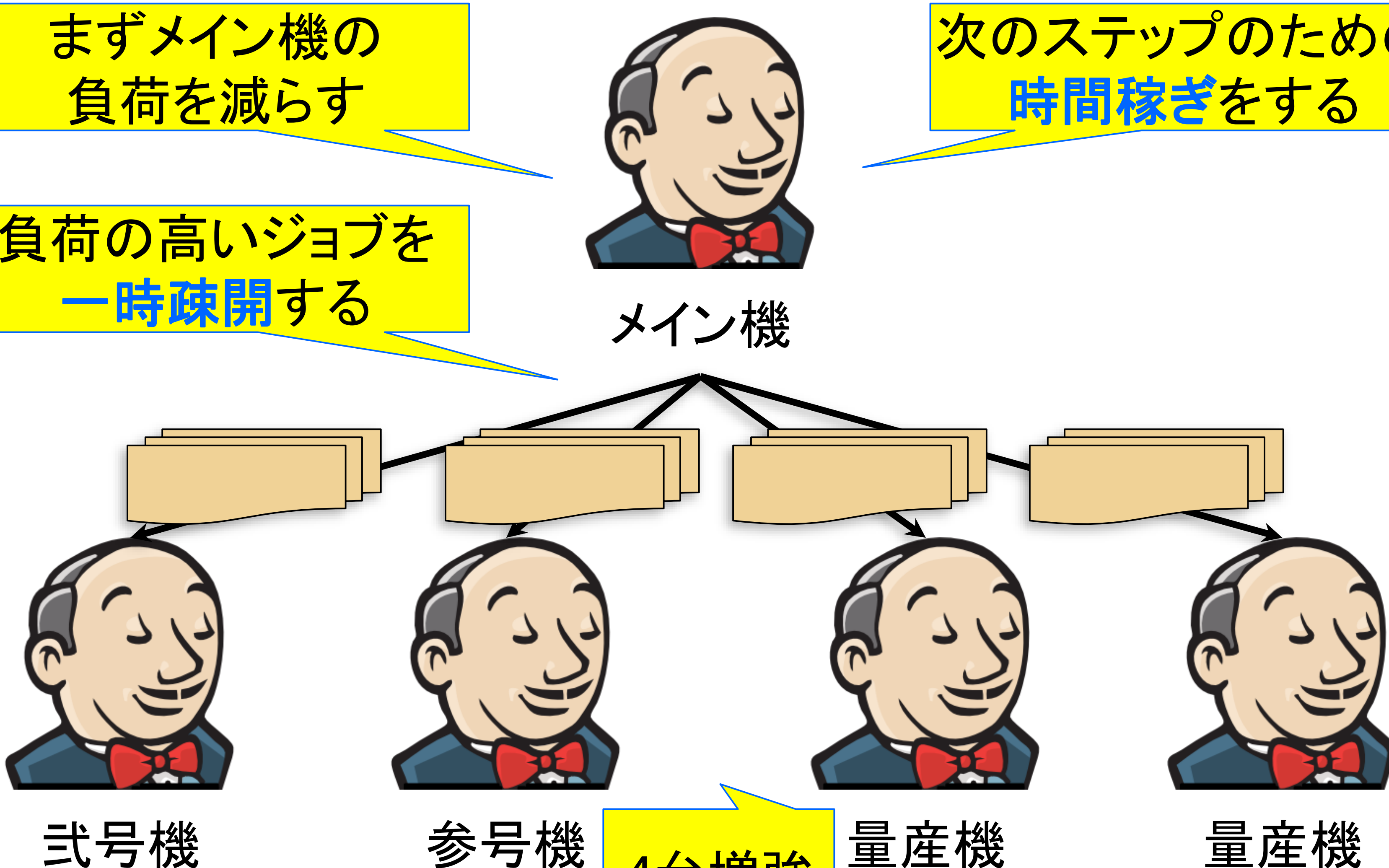
施策

(1) 単純にCIサーバ数を増やす

まずメイン機の
負荷を減らす

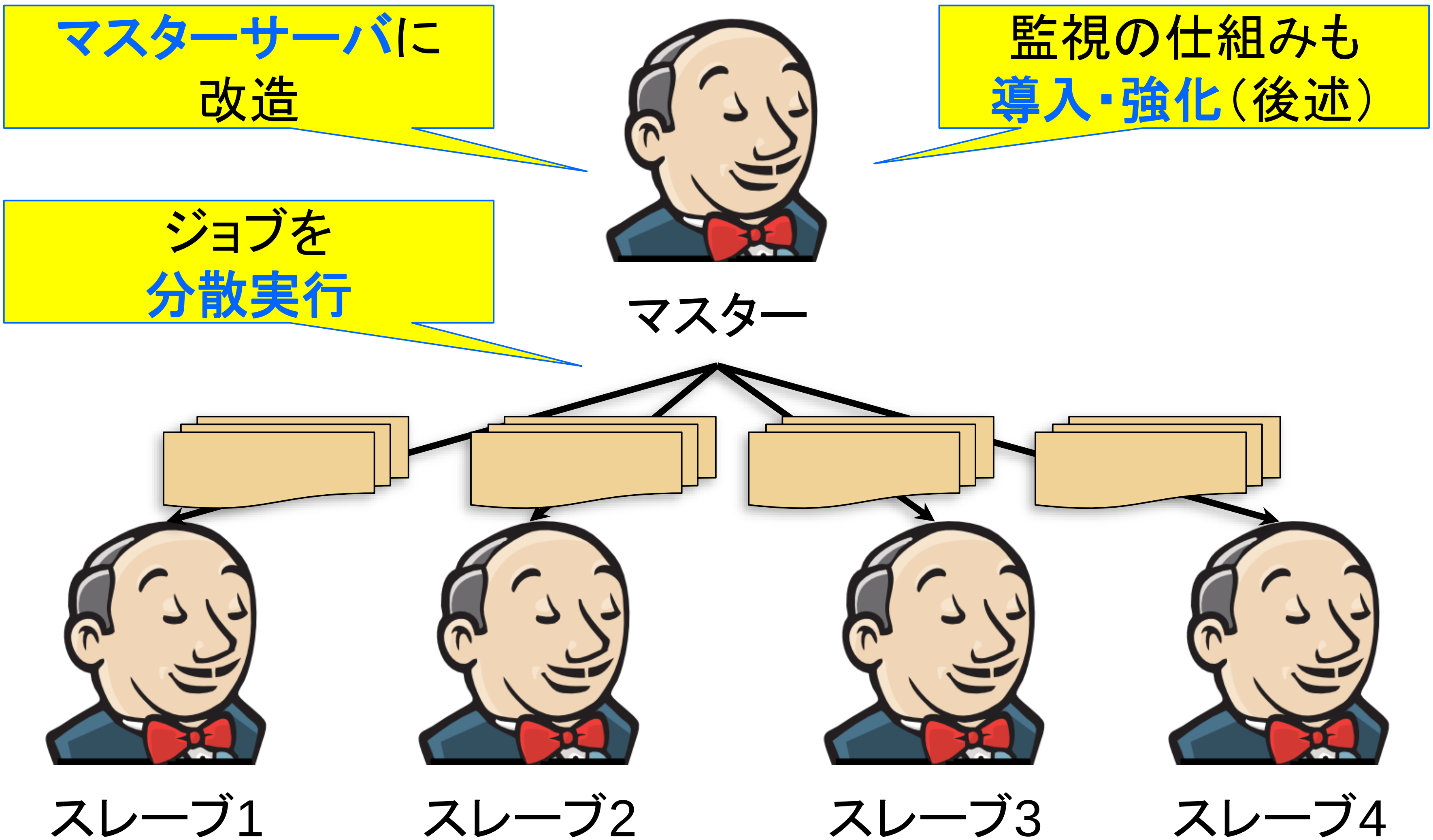
次のステップのための
時間稼ぎをする

負荷の高いジョブを
一時疎開する

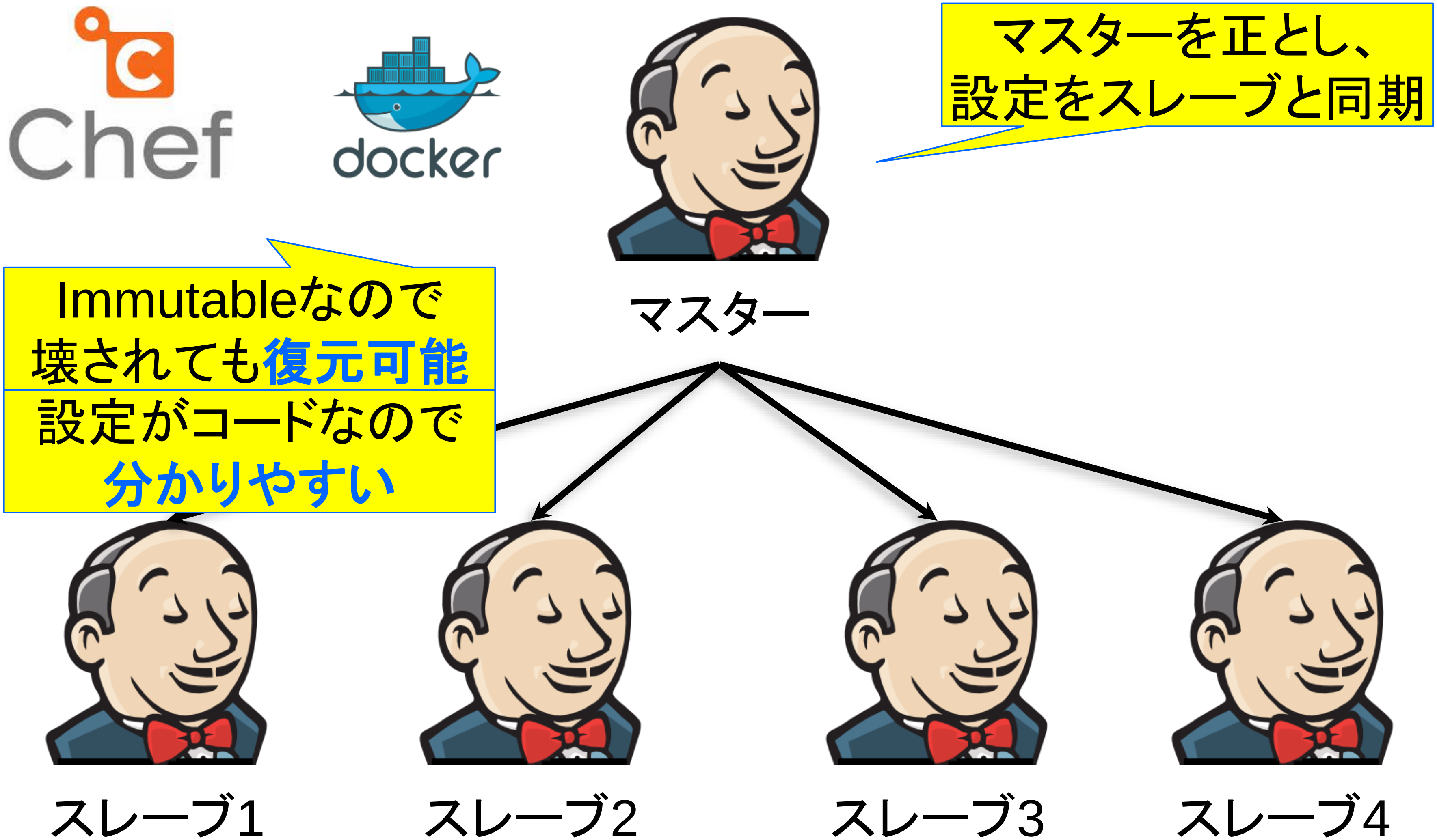


4台増強

(2) マスタースレーブ構成を適用する



(3) Infrastructure as Codeを適用する



成果

インフラの整備状況の予実

実施した施策	予定	実績
単純にCIサーバ数を増やす	1ヶ月	**1週間**
マスタースレーブ構成を適用する	1〜2ヶ月	**1週間**
Infrastructure as Codeを適用する	2ヶ月	**1ヶ月**

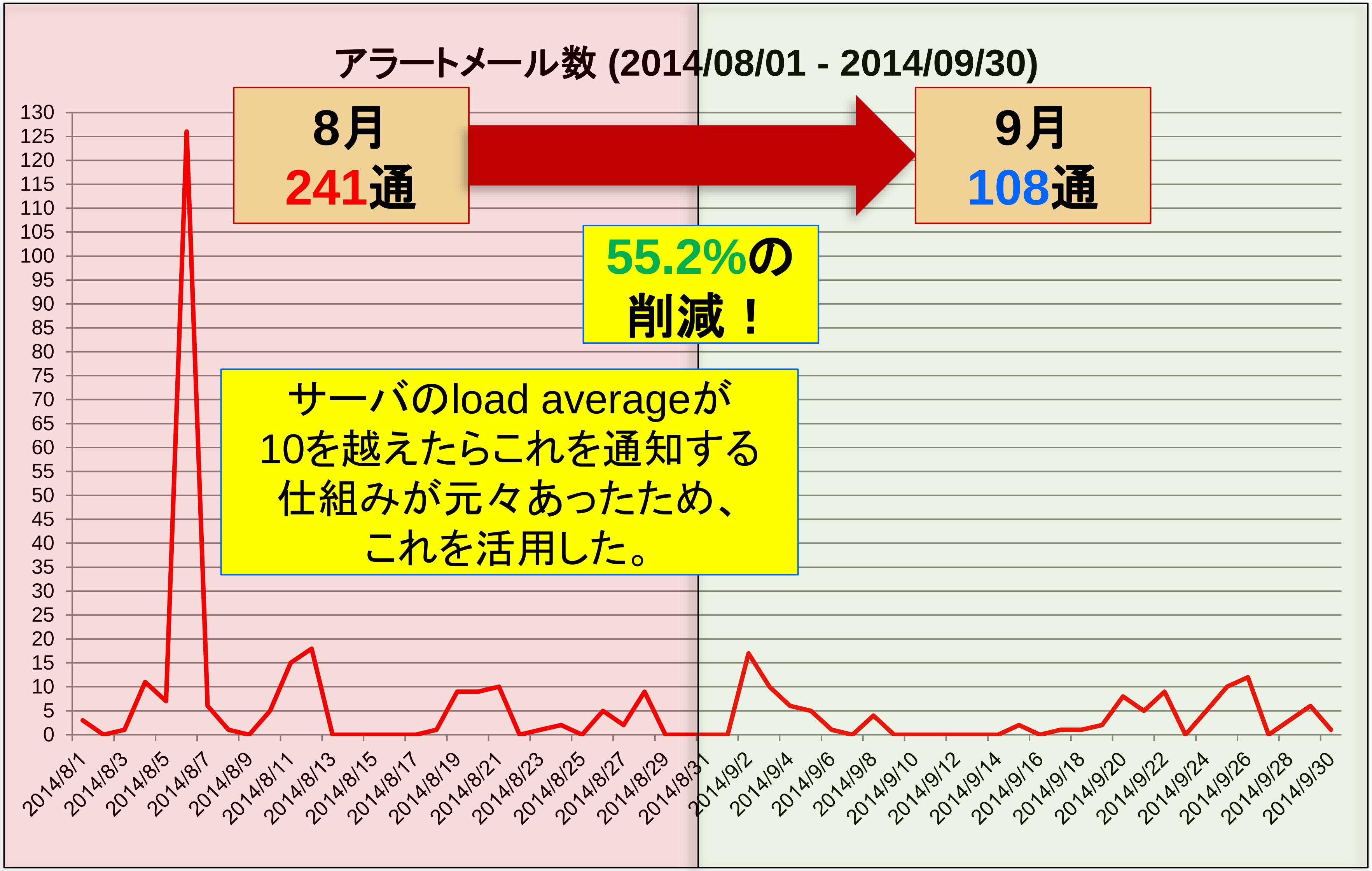
[要因]

- **直井さん主導**による、インフラ担当者との迅速な折衝
- **若者の人間離れ**（≒メンバーの予想以上のがんばり）

直井さんによる交渉・施策のポイント

- インフラとの交渉は、
  施策実現のために
  必要なことをやったに過ぎない。
- この時点で、次への布石を
  着実に打っておいた！

アラートメール数の推移にみる負荷減少



Infrastructure as Codeの有益性 (1)

採用前	採用後
1週間	30分

[要因]

- コミュニケーションミスによる依頼の往復を、
  ほぼ無くすことができた
- エンジニアが動作確認したものをインフラ担当者が実施するだけなので、**作業自体が容易になった**
- 万が一トラブルが起きても、
  Immutable Infrastructureなので**簡単に復旧できる**

Infrastructure as Codeの有益性 (2)

以前のインフラ運用体制

madokaグループの
homuraユーザで、
Tomcatを起動・停止
できるようにしたい

エンジニア

startup.shの権限を、
774から777に変更してほしい

インフラ担当

startup.shの権限を、
777に変更しました

shutdown.shが実行できない！
homuraユーザがmadokaグループに入っていない！

Infrastructure as Codeの有益性 (3)

以前のインフラ運用体制

コミュニケーションミスが
多発しがち

やり取りが多くなりがちで
時間がかかる

エンジニア

個々のサーバで
構成が異なることが多いため、
事前確認で漏れを無くすことが**困難**

実際に動作確認するまでは
安心できない

大概、依頼内容および設定結果に
誤りがある

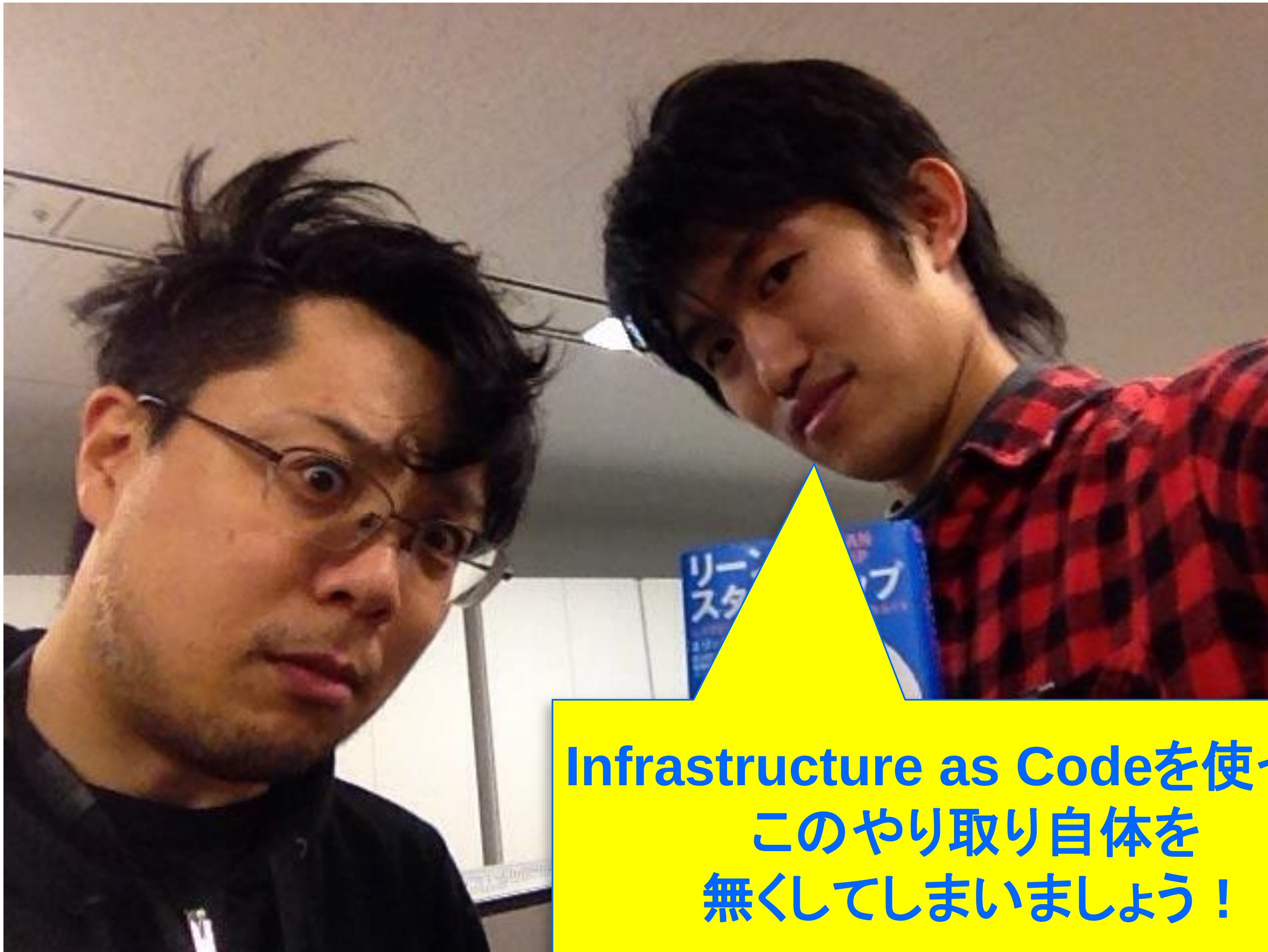
インフラ担当

なぜその作業を依頼されたのか、
その意図・目的までは**伝わりづらい**

指示された通りの設定は
（大体）できる

大概**多忙**なので、
作業が**後手にまわりがち**

実際に面倒な目に遭いました…



Infrastructure as Codeを使って、
このやり取り自体を
無くしてしまいましょう！

Infrastructure as Codeの有益性 (4)

新しいインフラ運用体制

Chef docker

エンジニア

インフラ担当

依頼前に各自のPCで
構築・動作確認が出来る

必要な全ての設定は、
Recipeとテストコードとして残せる

Recipeとテストコードの
Pull Requestで、
手戻りの少ない作業依頼が出来る

Recipeとテストコードを
実行すれば良い

ダメだったら**設定を戻せばよい**ので、
作業上の**リスクが少ない**

（再掲） インフラの整備状況の予実

実施した施策	予定	実績
単純にCIサーバ数を増やす	1ヶ月	**1週間**
マスタースレーブ構成を適用する	1～2ヶ月	**1週間**
Infrastructure as Codeを適用する	2ヶ月	**1ヶ月**

マスターサーバが軽くなったので、もう一工夫を。

「もうひと工夫」の重要性

メール1つ取っても、
1つ1つ着実に改善を続けたことの積み重ねが、成果を出せる組織の土台になっていく。

アラートメール (1) :改善前

09:40:04 up 154 days, 19:14, 2 users, load average: 11.59, 11.32, 7.16

[Heavy Process Top 10] CPU(%), Proccess
165 xxxxx

そもそもスペルが間違っているぞ！

- uptimeコマンドを使っているが、情報が冗長
- CPU使用率の高いプロセスは特定できるが…
  - プロセスのユーザは特定できない
  - 何が問題でどう解決すれば良いのかの情報が少ない
- メモリ使用率の高いプロセスは分からない
- どのジョブが原因なのかを特定できない

アラートメール (2) ：改善開始

Load average: (1.03, 1.05, 1.23)
--- Top 10 Heavy CPU consumers ---
tomcat xxxxx
--- Top 10 Heavy Memory consumers ---
ito xxxxx
--- Jobs currently running ---
- Sayaka

- uptimeコマンドの実行結果を、**load averageのみに絞った**
- CPU使用率に加え、**メモリ使用率の高いプロセス**も特定・出力するようにした
- 当該プロセスの**ユーザ**を特定・出力するようにした
- 原因となる**ジョブ名**を特定・出力するようにした
  但し、一番下に出力しているため、
  都度メールを一番下まで見なければならない

アラートメール (3) : 更なる改善

Load average: 10.0, 5.2, 2.4
--- Jobs currently running ---
- Sayaka
--- Top 10 Heavy CPU consumers ---
tomcat xxxxx
--- Top 10 Heavy Memory consumers ---
ito xxxxx

原因となるジョブ名を、**メールの最初に出力**するようにした

- ジョブの特定が非常に容易になった
  （メールを開けば一目で分かる）
- 問題のあるジョブの特定と原因調査が、非常に容易になった
  - **と言うよりも、この時点で初めてこれができるようになった**

（特に負荷の高い）ジョブの実行頻度とその推移

ジョブ名	アラートの頻度
Madoka	32
Homura	17
Sayaka	15
Anko	13
Mami	13
Charlotte	4
QB	3
Hitomi	2
Uro-Buchi	2

負荷の高いものから
（スレーブへの移行などの）
対応を優先的に行なう

これを**週次で**関係者へ
メール通知することで、
自発的な活動を促す

失敗から学ぶアジャイル

最初から全情報を計測するのではなく、
出来るところから徐々に集めていけば良い

全部が全部役に立つ情報ではない
→役に立つ情報だけを計測すれば良い

プロジェクトメトリクスを定期的に振り返り、
適宜見直しを行なう必要がある

想定外の状況が生じることもままある
→積極的に活用することをまず考えよう

電撃的勝利！

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー･チームを育成していくか？

5. 結論

2014年9月下旬

楽天テクノロジーカンファレンス
絶賛準備中の時期ですよ。

成果の報酬としての新たなタスク

部の2014年度（～12月）の目標として、
全プロダクトのUTのラインカバレッジを
65%以上にすることが設定されていた

↓

CIサーバの負荷軽減・改善が
想定以上に速く進んだため、
この目標もスコープに含めることとした

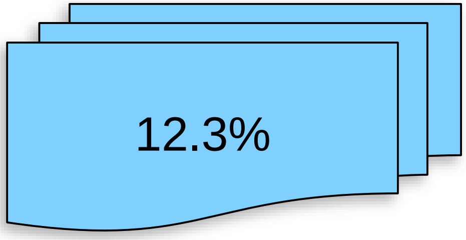
UTカバレッジを目標に設定した狙い

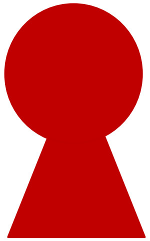
UTカバレッジに関する当時の課題

UTカバレッジの
計測手段と
メトリクスの不足

ツールの乱立

- Cobertura
- Emma(JaCoCo)
- djUnit…





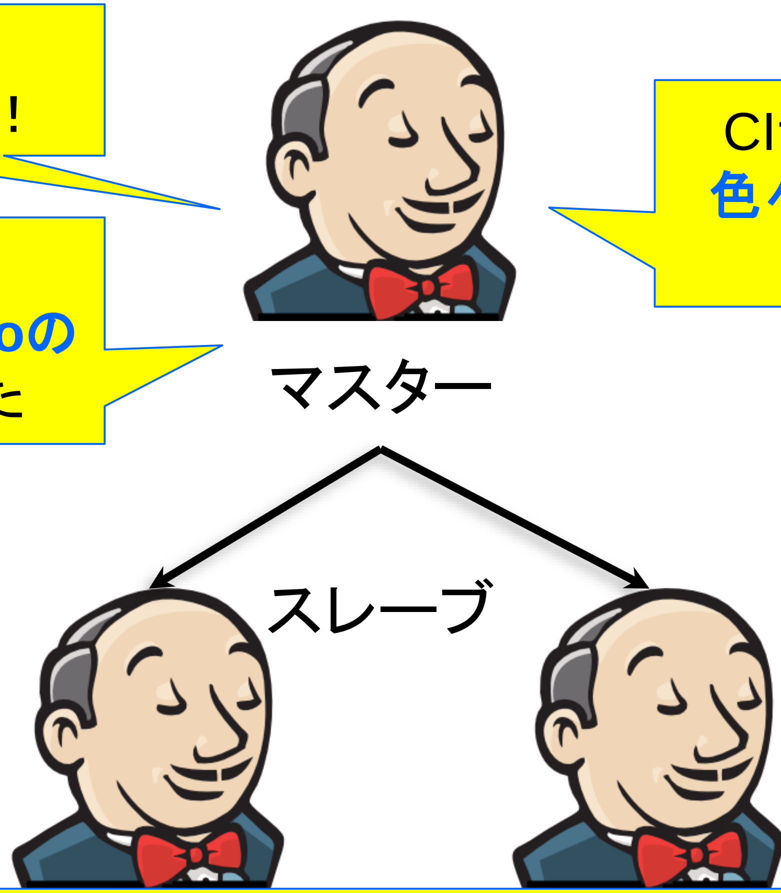
施策をリードする
メンバーの不足

(一方で)マスターサーバの更なる負荷軽減の効果

Viewを
使えるようになった！

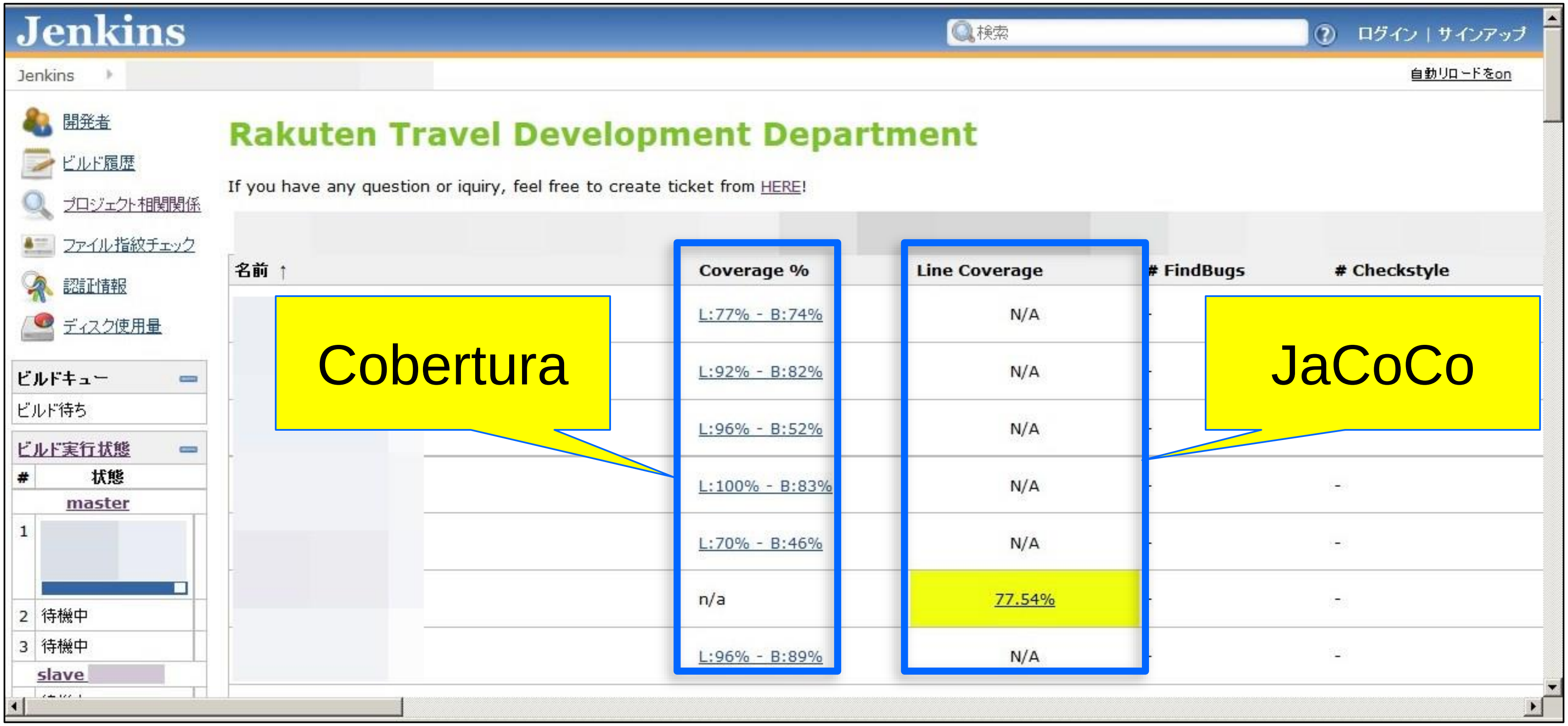
ツールを統合し、
CoberturaとJaCoCoの
2つに絞ることとした

CIサーバを自発的に
色々と試すメンバーが
増えてきた



次のステージに移れる！

(1) Viewを作成し、カバレッジを常に誰でも確認可能にした



(2) カバレッジの推移を、週次で記録・報告するようにした

計測対象ジョブ数	74
うちカバレッジを取得している数	67
目標達成済ジョブ数	35
達成率	47.3% (+18.9% WoW)

先週比を追加し、
成長の共有と
成果の強調を図った

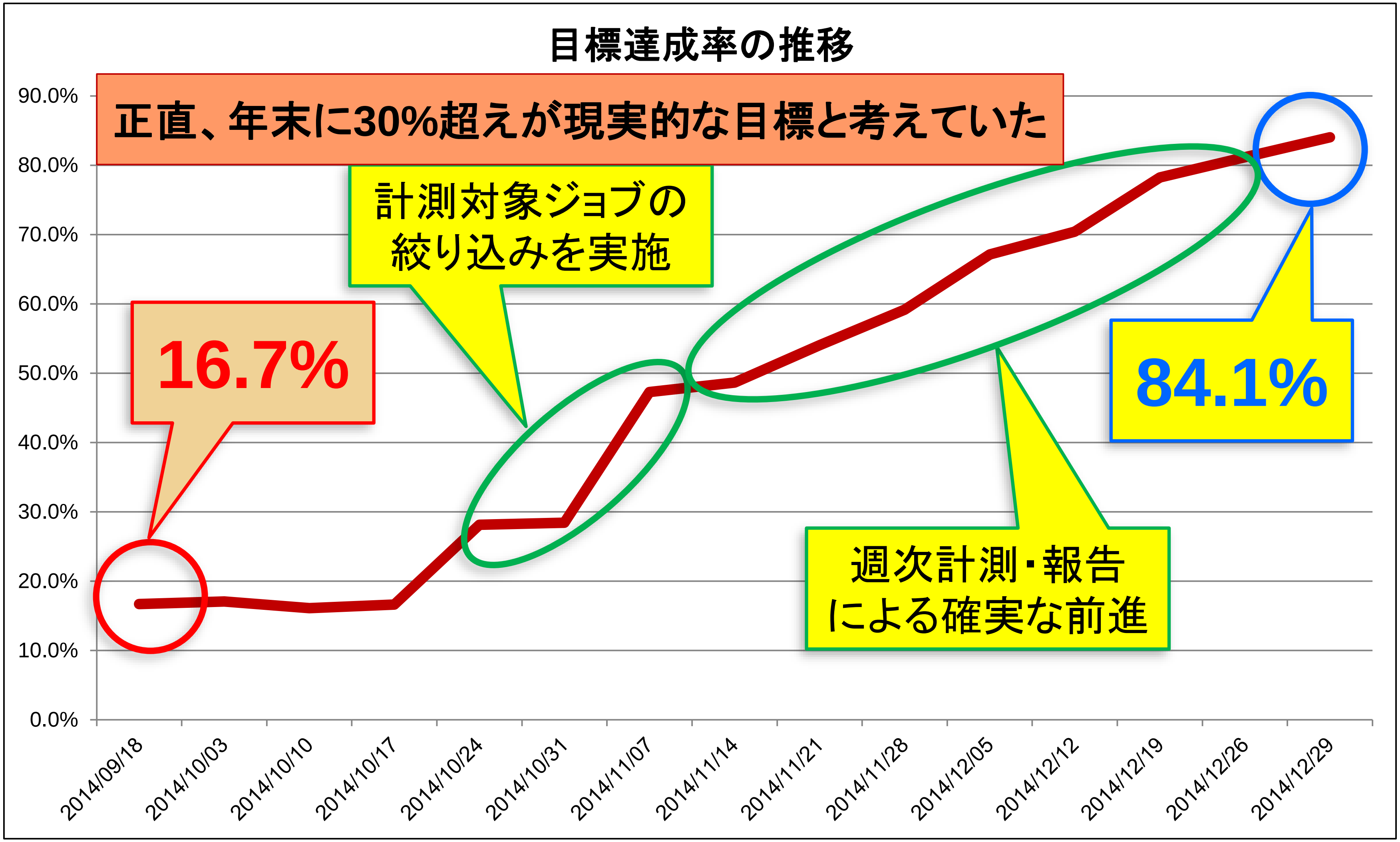
(3) 各グループからメンバーを徴発し「65%チーム」を組織した

対象グループ：**約10**

CIサーバを**自発的に色々と試すメンバー**を優先的にアサインした

このチームを介して、**UTカバレッジ向上の効率化を目指した**

成果としてのUTカバレッジの目標達成率の推移



実際に成果を出してみて

数値をキチンと計測する
ということから始めないと、
そもそも改善なんて
生まれませんよ！

完全勝利！！

1. 課題の整理

2. 何を持って成果とすべきか？

3. どうやってCIサーバの負荷を減らすか？

4. どうメンバー･チームを育成していくか？

5. 結論

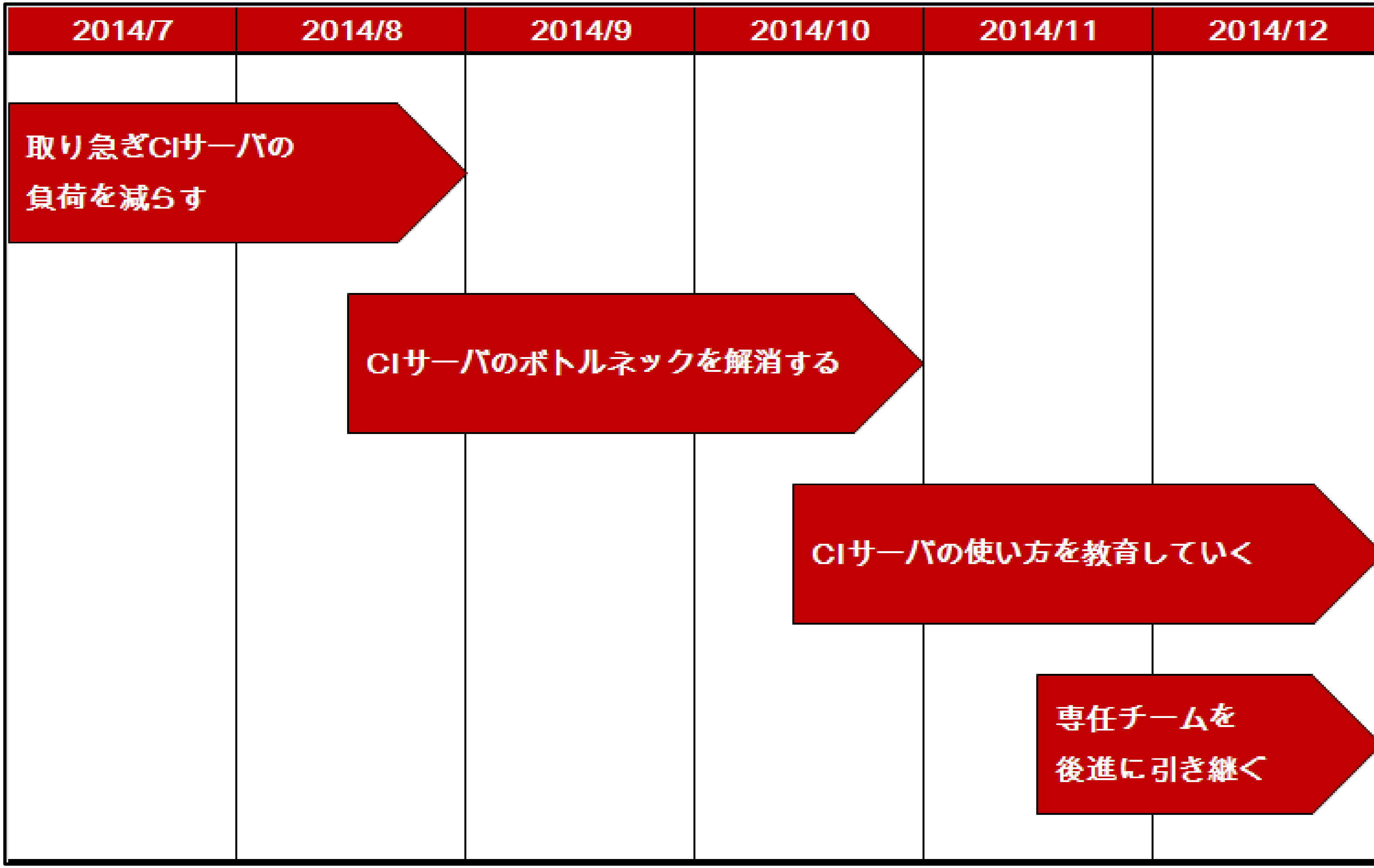
最終的な成果

リリース遅延を解決し、
再発しないようにした

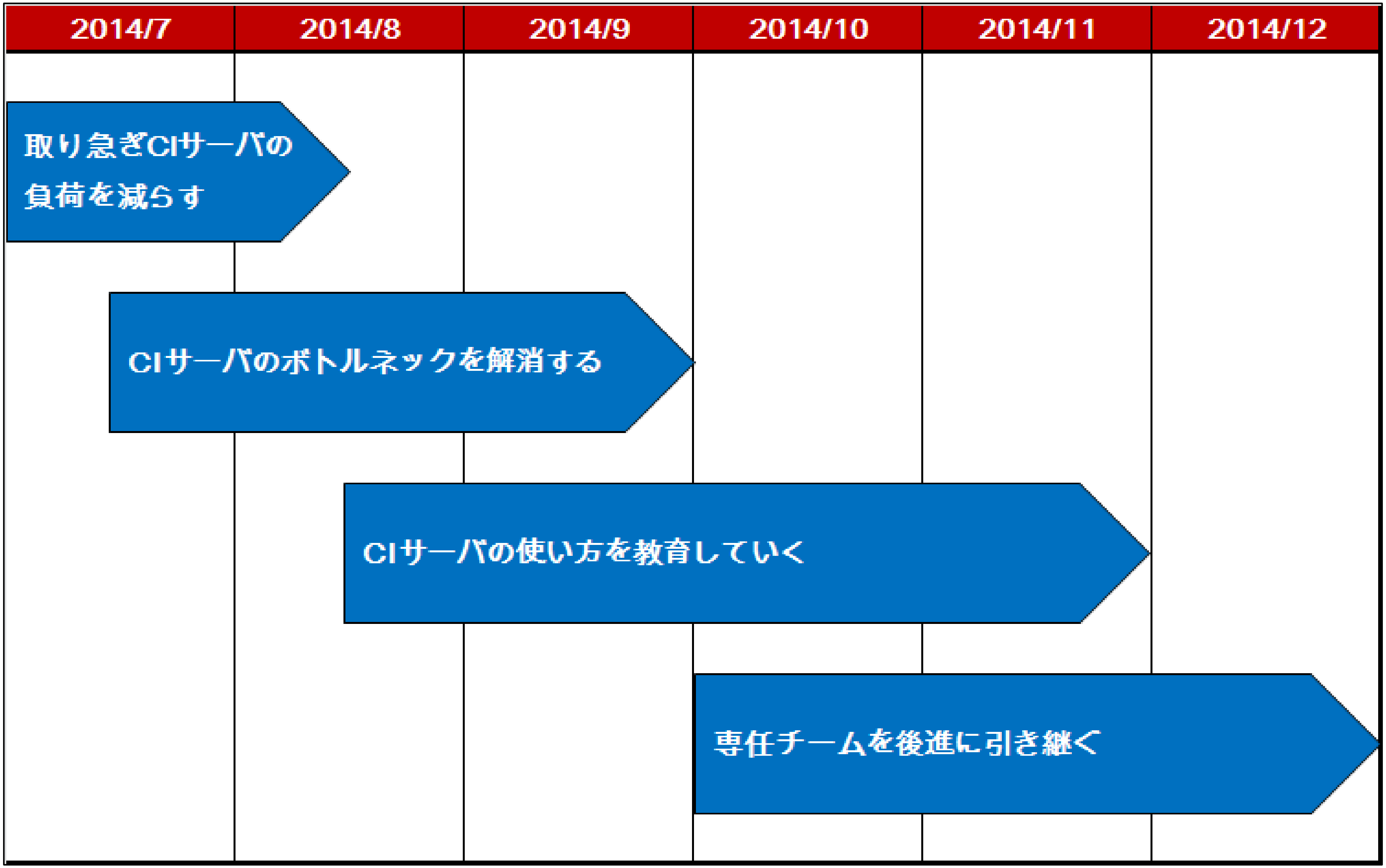
CIサーバのスケールアウトを実現
・550前後のジョブが現在稼働中

後進を育成し、
専任チームそのものを引き継いだ

ロードマップ(予定)


2014/7
2014/8
2014/9
2014/10
2014/11
2014/12
取り急ぎCIサーバの負荷を減らす
CIサーバのボトルネックを解消する
CIサーバの使い方を教育していく
専任チームを後進に引き継ぐ

ロードマップ（実績）


2014/7
2014/8
2014/9
2014/10
2014/11
2014/12
取り急ぎCIサーバの
負荷を減らす
CIサーバのボトルネックを解消する
CIサーバの使い方を教育していく
専任チームを後進に引き継ぐ

残課題

IT/STレベルのテストの自動化がまだまだ不足している

テスト環境でデータ競合が生じる

- Immutable Infrastructureで…

リリース自動化がまだ不十分

- Blue-green deploymentを…

新たな課題

- UTカバレッジを上げさえすれば品質も上がるだろうという
安易な幻想を、
一部マネージャに持たせてしまいそうになった。
- 改善が劇的だったがゆえに、
簡単に改善ができるものと勘違いされがち。

成功につなげるアジャイル

戦略と(特に自動化)技術との融合による
高速なPDCAサイクルの実現

PDCAサイクルの有効性を高めるための
プロジェクトメトリクスの**発見・計測・見直し**

上記の組み合わせによる
人材・組織育成とコラボレーションの拡大

プロジェクトメトリクスのポイント

考えることのプラスになる情報を見つけよう

- 現状の問題はどこにあるのか？
- 改善施策の効果はあったか？

数値の変化に意味を見いだそう

- 数値の変化にこそ、真の知識が詰まっている
- 変化が見える情報を見つけることが勝利の鍵

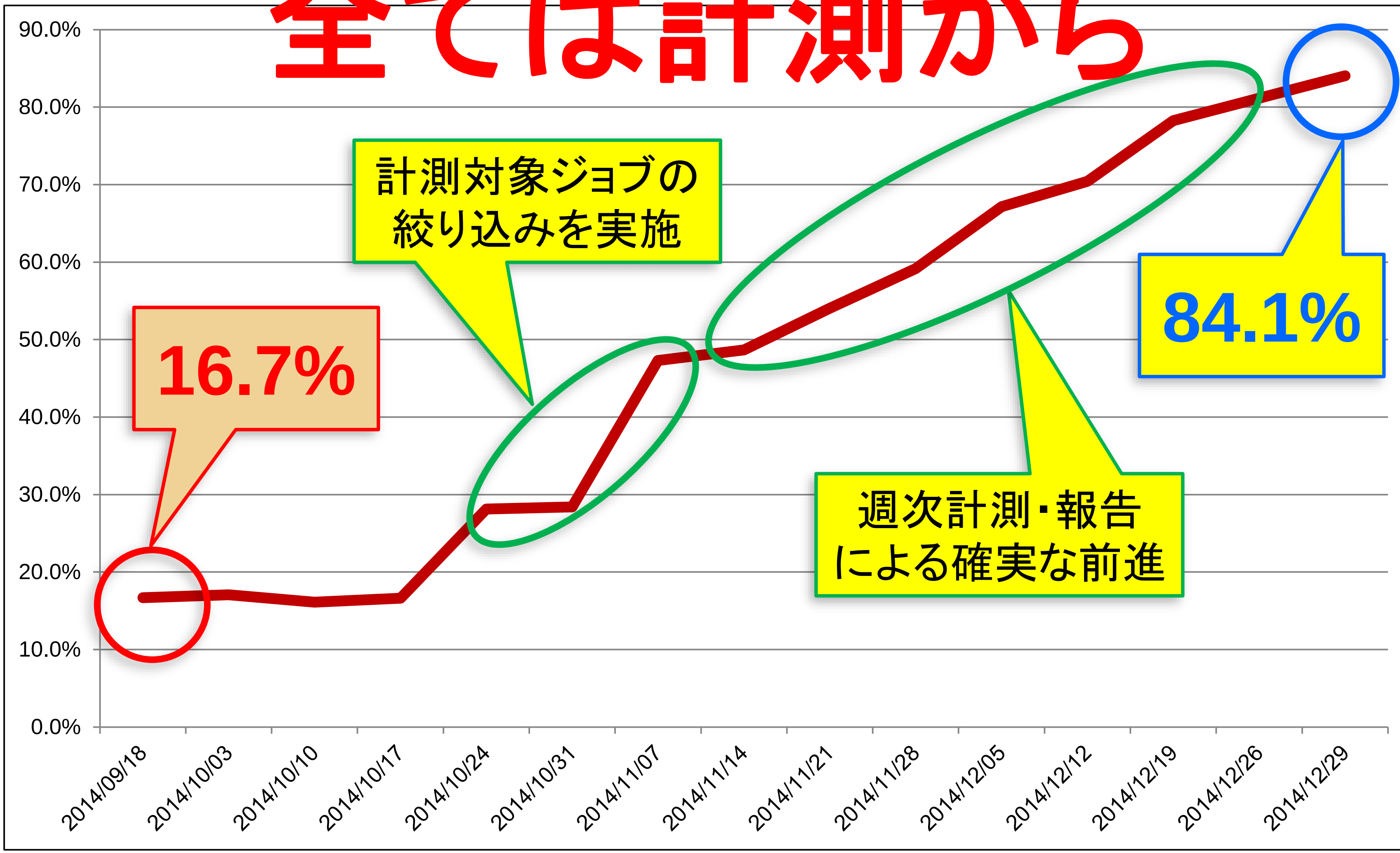
メトリクスを定期的に振り返り改善していこう

- 役に立たなければ、潔く捨てる覚悟も必要
- 足りなければ創ろう

コミュニケーションの手段として活用しよう

- 今まで分からなかった情報を元に話をしてみると、更なる発見が得られる

全ては計測から


90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
2014/09/18
2014/10/03
2014/10/10
2014/10/17
2014/10/24
2014/10/31
2014/11/07
2014/11/14
2014/11/21
2014/11/28
2014/12/05
2014/12/12
2014/12/19
2014/12/26
2014/12/29
16.7%
計測対象ジョブの
絞り込みを実施
週次計測・報告
による確実な前進
84.1%

滾らせろ！