SREの文化と組織

Infra Study Meetup #3

古川 雅大

2020/06/16

自己紹介

名前: 古川 雅大 (id:masayoshi Twitter: @yoyogidesaiz)
所属: 株式会社はてな

Mackerel*開発チーム内のSREチームでSREs
同チーム内のSREテックリード

* Mackerel https://mackerel.io/

はじめに ~ infra study meetup #1, #2 ~

Infra Study Meetup #1

IaCに限らず、新しい技術用語が出てくると、関連するツールやプロセスが注目され、それらを導入すれば課題は解決する、と思われがちです。ですが、大切なのはその言葉の背後にある思想や哲学を理解し、状況に合わせて適切にツールやプロセスを選択することです。

Infra Study Meetup #2

我々は正しくクラウドネイティブを実現するためにも、その思想を正しく理解した上でKubernetesなどの技術スタックを選択していく必要があります。思想を正しく理解しないまま選択することは、こういった技術スタックをVMの代わりとして使ってしまいかねません。

Infra Study Meetup #1「Infrastructure as Code」https://forkwell.connpass.com/event/171560/
Infra Study Meetup #2「VM時代の開発とCloud Native時代の開発」https://forkwell.connpass.com/event/173289/

はじめに ~ infra study meetup ~

- 思想や哲学を理解して、適切にツールやプロセスを選択出来るようにする
  - 思想、哲学、ストーリー、Why
  - ツール、プロセス、How
- (発表者が考える) infra study meetupの思想
  - 「これまで」をより理解する、整理する
  - 「これから」に対応する、作り出す

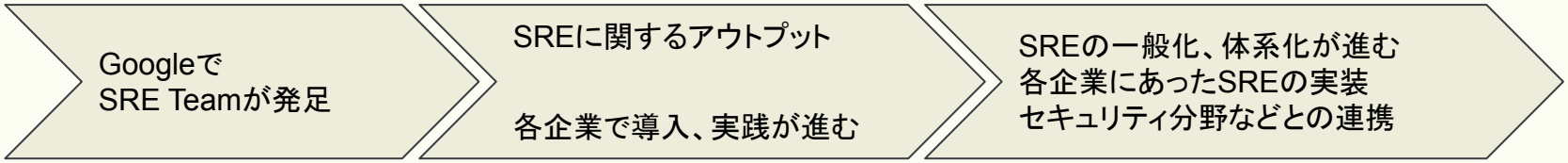
はじめに ~ 今日の発表内容 ~

- SREのこれまでとこれから
- SREとは
- SREの文化
- SREと組織
- 付録 (時間の関係で発表では触れない内容)

※ 全て発表者が現段階 (2020/06)で考えた内容です
正直に言うと実際のところはよくわからないのだ

SREのこれまでとこれから

SREのこれまでとこれから



- これまで
  - 「一企業(Google)のプラクティス」から「Webサービス事業者のプラクティス」へ
  - 「企業のプラクティス」から「研究分野」へ
- これから
  - 企業や組織毎にあったSREの実践、実装の登場
  - セキュリティ分野など他の分野との連携
  - 産学連携やエンジニアと研究者の連携による新たな手法の発見

(参考) SRE NEXT 2020[*1]

- 発表の割合
  - 特定の技術に関する話題と組織、コミュニケーションの話題半々ぐらい
- 「現場で課題に感じている事」会場アンケート結果[*2]
  - 組織とマネージメント
  - Toilの撲滅

*1 SRE NEXT 2020 https://sre-next.dev/
*2【SRE NEXT】Closing Talk https://www.youtube.com/watch?v=pJXT4qwg5Lc

SREの「これから」に対応するために

- 組織やコミュニケーション、文化の話は不可欠で難しい話題
- 「SREとは」を再整理し、自分たちの組織にあった形で進められるようにしていく

SREとは

SREとは

“SREとは、ソフトウェアエンジニアに運用チームの設計を依頼したときにできあがるものです。”

Benjamin Treynor Sloss
(Google SREの創始者)

Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy 編, 澤田 武男, 関根 達夫,細川 一茂,矢吹 大輔 監訳、Sky株式会社 玉川 竜司 訳, ”SRE サイトリライアビリティエンジニアリング――Googleの信頼性を支えるエンジニアリングチーム”, オライリー・ジャパン, 2017

SREとは?

“Class SRE Implements DevOps”

“SRE vs. DevOps: competing standards or close friends?” https://cloud.google.com/blog/products/gcp/sre-vs-devops-competing-standards-or-close-friends

SREとは??

- SRE = SLI/SLOによる意思決定？
- SRE = ソフトウェアエンジニアリングによる自動化？
- SRE = システムのモニタリング？
- SRE = DevOpsの実装？
- SRE = インシデント対応？
- SRE = チームマネージメントによるToilの撲滅？

SREとは???

“それでは、サイトリライアビリティエンジニアリング(SRE)とはいったい何なのでしょうか。この名前が、その内容をはっきりとは表現できていないことは認めざるをえません。”

“SRE サイトリライアビリティエンジニアリングはじめに”より

“私達が学んだことを他の組織も利用できるようにすることと、SREという言葉が意味する役割と意味をもっと上手く定義することの両方を目的としている”

“SRE サイトリライアビリティエンジニアリングはじめに”より

Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy 編, 澤田 武男, 関根 達夫,細川 一茂,矢吹 大輔 監訳、Sky株式会社 玉川 竜司 訳, ”SRE サイトリライアビリティエンジニアリング――Googleの信頼性を支えるエンジニアリングチーム, オライリー・ジャパン, 2017

SREを紐解く

- SREはシステムの信頼性に焦点を当てる
  - これはSREの内容? => 信頼性に関わることならYes
- 「信頼性こそがあらゆるプロダクトの基本的な機能」だから[*1]
  - 誰も使えないシステムは、有益なものではない

*1 Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy 編, 澤田 武男, 関根 達夫,細川 一茂,矢吹 大輔 監訳、Sky株式会社 玉川 竜司 訳, ”SRE サイトリライアビリティエンジニアリング――Googleの信頼性を支えるエンジニアリングチーム”, オライリー・ジャパン, 2017

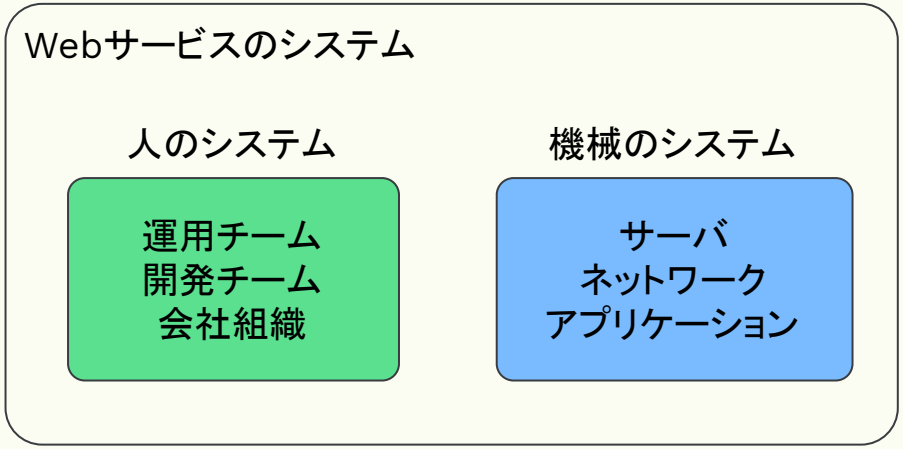
SREを紐解く ~信頼性とは~

- 信頼性とは
  - 「システムが求められる機能を、定められた条件のもとで、定められた期間にわたり、障害を起こすことなく実行する確率」[*1]

*1 P. O'Connor and A. Kleyner, Practical Reliability Engineering, 5th edition: Wiley, 2012.

SREを紐解く ~システムとは~

- Webサービスには「人のシステム」と「機械のシステム」がある
- Webサービスの信頼性は、2つのシステムを考慮する必要がある



SREを紐解く ~定められた条件とは~

- (SREの文脈では多くの場合)ビジネス的な条件
  - 限られたコスト (人、金、時間)
  - 売上の増加 (魅力的な新機能の開発)
  - 変化への対応 (スケーリング)

スケーリングとは機械的なことよりも、むしろビジネスプロセスのスケーリングに関することです。

“SRE サイトリライアビリティエンジニアリングはじめに”より

Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy 編, 澤田 武男, 関根 達夫,細川 一茂,矢吹 大輔 監訳、Sky株式会社 玉川 竜司 訳, ”SRE サイトリライアビリティエンジニアリング――Googleの信頼性を支えるエンジニアリングチーム”, オライリー・ジャパン, 2017

発表者の考えるSREとは

- 信頼性のあるWebサービスを提供するために、
  「条件」を考慮した「システム」を設計、実装、運用の課題を解決

- 「条件」とは
  - (主な条件は)ビジネス上の条件 (コスト、機能の開発、変化)

- 「システム」とは
  - 機械的なシステム (サーバー、ネットワークなど)
  - 人のシステム (運用チーム、開発チーム、会社全体など)

SREの文化

SREの文化

- 文化という切り口で紹介
  - プラクティスや原則といった切り口は既にある
- SREという単語やプラクティスがなかったと仮定して、どういった考え方、文化を持っていれば現在のSREを生み出すことが出来るか?
  - 文化を習得すれば応用し新たなプラクティスを生み出すことが出来る
- とはいえ、発表者もまだ整理中で完成度は低いかもしれない
  - こういった見方もあるんだなぁぐらいの気持ちで

SREの文化 ~ 人と機械のシステム ~

- Webサービスは以下の2つで成り立つ
  - 開発運用する人の組織(システム)や利用するユーザ
  - 機能を提供するアプリケーション、インフラなどのシステム
- 開発チームと運用チームでの信頼関係、ユーザとの信頼関係
  - 目標、意思決定の基準、考え方を共有し※ 信頼関係を築いていく
- インフラやアプリケーションの信頼性
  - 技術で殴る！

※ この目標、基準、考え方の共有をユーザとするには?という課題に更に踏み込むとCREになっていくのではないか

SREの文化 ~「信頼性」にフォーカスする ~

- 判断基準が「信頼性」であること
  - 例: SLI/SLO, SLO Docs, エラーバジェットポリシーに従った意思決定
- インシデント対応, On Call対応
- 「変化」など信頼性が落ちる事項への対応
  - 例: リリースエンジニアリング
- 人の心理やチーム間とのコミュニケーション方法、信頼関係
  - 例: 批難のないポストモーテムの文化
  - 例: 障害時などストレス配下にある状態での心理状態
- 他分野の信頼性に関する知見も積極的に取り込む
  - 信頼性工学、安全工学、組織論、心理学

SREの文化 ~ 完全を目指さない ~

- 信頼性100%を目指さない
  - 例: SLI/SLOを元にした設計
- Toil, 割れ窓, 割り込みタスクを0%に出来るわけではない
  - 例: Toilを50%以下に維持する
- オペレーションを全てなくすことは出来ない
- 1回のチャレンジで完全を目指さない
- 1人で全て出来る必要はない

SREの文化 ~ 信頼性とのトレードオフを理解 ~

- 変更速度と信頼性
  - 変更速度が早ければ信頼性が下がります
- 複雑性(機能性)と信頼性
  - 複雑(多機能)であればあるほど信頼性が下がります
- 変化量と信頼性
  - 変化が大きければ信頼性が下がります

SREの文化 ~ エンジニアリング ~

- ソフトウェアエンジニアリング
- システムエンジニアリング *
- 信頼性に関する課題の解決
  - 必ずしもエンジニアリング = コードを書くことだけではない
  - 例: “トレードオフ”をモデルに落とし込み定量分析可能にする
  - 例: SLOの決定、SLO Docs

* “The Systems Engineering Side of Site Reliability Engineering” https://www.usenix.org/publications/login/june15/hixson

SREと組織

SREと組織 ~ SREをどうやって導入するか ~

- 結論
  - SRE Workbookなどを参考にSLI/SLOなどのプラクティスを組織にあった形で、少しづついい感じに進めて下さい！
- あなたがSREを組織に導入をするとして
  - 先ほど紹介したSREの定義とSREの文化を使って考える
  - 社内にSLI/SLOを導入したときの例を紹介

SREと組織 ~ 導入準備 ~

- 組織もシステム
  - 組織にSREという大きな機能を実装すると考える
- システムに変更を加えると信頼性が下がることに注意し、段階的に入れていく※
  - 変更量と信頼性　(少しづつ導入しましょう)
  - 変更速度と信頼性 (導入には時間がかかります)
  - 複雑性と信頼性　(最初はシンプルでよいでしょう)
- 最初から完全を目指さない
  - 仮置きの目標や、最初の設計を頑張りすぎない

※ 信頼性が下がる = SREは幻想だ、使い物にならない、大変だ、ストレスに感じる、やりたくないと思われる確率が上がる
余力があるなら組織に関する変更の SLI/SLOやエラーバジェットに似た基準を作ってみても良いかもしれません

SREと組織 ~ 導入準備 ~

- 組織が変更容易な組織か?
  - 組織の規模
  - 組織に対する変更手順が整備されているか(リリース手順)
  - 一部の部署で試すことが出来るか(カナリアリリース)
  - 足りなければ組織のリリースエンジニアリングを優先したり、並行して進めることを検討
- 組織が挑戦可能で失敗を批難しない文化か?
  - 心理的安全性があるか?
  - 挑戦を評価してくれるか?

SREと組織 ~ 誰に説明するか ~

- 信頼性のあるWebサービスを提供するために、
  「条件」を考慮した「システム」を設計、実装、運用の課題を解決
- 「(ビジネス上の)条件を決定出来る人」へ※SREを説明する必要がある
  - 自分 (やりやすい)
  - Product Manager, Product Owner
  - CTO
  - 経営陣
  - 別会社 (難しい)

※ SLI/SLOのプラクティスに従っていれば自然と SLOの決定権を持つ人に説明しに行くことになるでしょう

SREと組織 ~ 誰に説明するか ~

- 信頼性のあるWebサービスを提供するために、
  「条件」を考慮した「システム」を設計、実装、運用の課題を解決
- システムに関係する人に※説明する必要がある
  - 開発チーム
  - SREチーム
  - サポートチーム
  - 営業チーム

※ SLI/SLOのプラクティスではSLO DocsのReviewerやエラーバジェットポリシーに関係するメンバー

SREと組織 ~ 何を説明するか ~

- SREの全てを説明するのは難しい
  - 最初は全てではなく一部を理解してもらえば良い(非エンジニアにはなおさら)
- SLI/SLOならば、SRE本の以下の章をメインにする(SRE Workbook 2章もおすすめ)
  - 3章 リスクの受容
  - 4章 サービスレベル目標
- ビジネスにとってSREは便利なツールであることを示す
  - データがあるなら示すと良い
    - 例: リリース当初の変更が盛んだった時期のアラート数と開発が落ち着いたときのアラート数

※ 今発表の資料にあるSREの文化の説明等はSRE初学者向けではない(設計者、開発者向けドキュメントに近い)

(参考)社内向け説明資料例

わかる！SRE！

id:masayoshi

この資料でわかってほしいこと

- SREにおけるSLOとエラーバジェットの考え方
  - 特に開発速度と信頼性のトレードオフをプロダクトオーナー,P,Dなどは理解してもらいたい
  -
- SREはいきなり完全な形で実践できるわけではないこと
  - 段階的にはすぐに始められるが、形になるにはそれなりの時間と工数を要する
  - SREsの人数も必要なので、各チームで段階的に進めていくことになる
  -
- 段階的な導入はSREサブ会で作成した「SRE成熟度レベル」を参考に
  - まだ初版ではあるが、SREをやっていく上での段階を定義している
  - チームでのSRE目標や、SLOの決め方、またSREs個人にお願いするタスクや成果目標の指針にしてほしい

開発と信頼性のバランスとは

- 機能を開発し、リリースすると信頼性が下がる
  - システムやサービスが複雑化し、バグなどでエラー率が増加
  - ユーザが増加し、パフォーマンス問題やエラー率が増加
  -
- 信頼性を高めようとすると、機能開発の速度が下がるor停止する
  - バグフィックスやパフォーマンスチューニングの工数増加により開発ができない

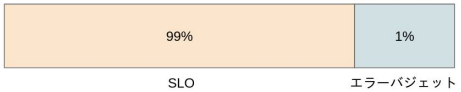
開発と信頼性はトレードオフの関係がある

SRE(Site Reliability Engineering)とは

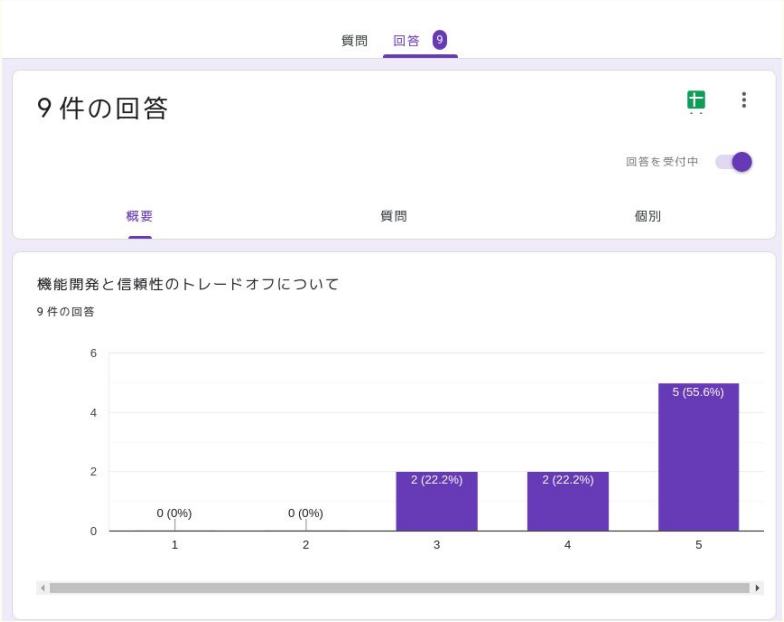
- Webサービスやサイトの信頼性を高める技術では**ない**
  - なぜなら、信頼性を高めることを目標にすると、機能開発を止めることが最善手になるから
  - 可用性を高めましょうという目標は、開発を止めましょうということにつながる
    - やりたいことはそうではない
    -
- サービス開発と信頼性の**バランスをとる**技術
  - サービスに必要な信頼性の中で、最大限に機能開発をやっていく
  - サービスに必要な信頼性が保てなくなりそうな状況を検知して、一時的に機能開発をやめてバグフィックス、パフォーマンスチューニングに工数を割く
    - こっちがやりたい

開発と信頼性のバランスはどうとるのか?

- SLOを用いて、定量的にバランスを取る
  - サービスに必要な信頼性をSLOとする (一番現状から近いものは可用性目標)
  - 100% - SLO = エラーバジェット(機能開発加速のための貴重な予算)とする
- エラーバジェットを使い切るまでは機能開発をする
- エラーバジェットを使い切ったら機能開発を止めて、バグフィックス、パフォーマンスチューニングなど
  - エラーバジェットを使い切ったら、何をすべきかを定めたものがエラーバジェットポリシー



(参考) 説明したらアンケートをとっておこう

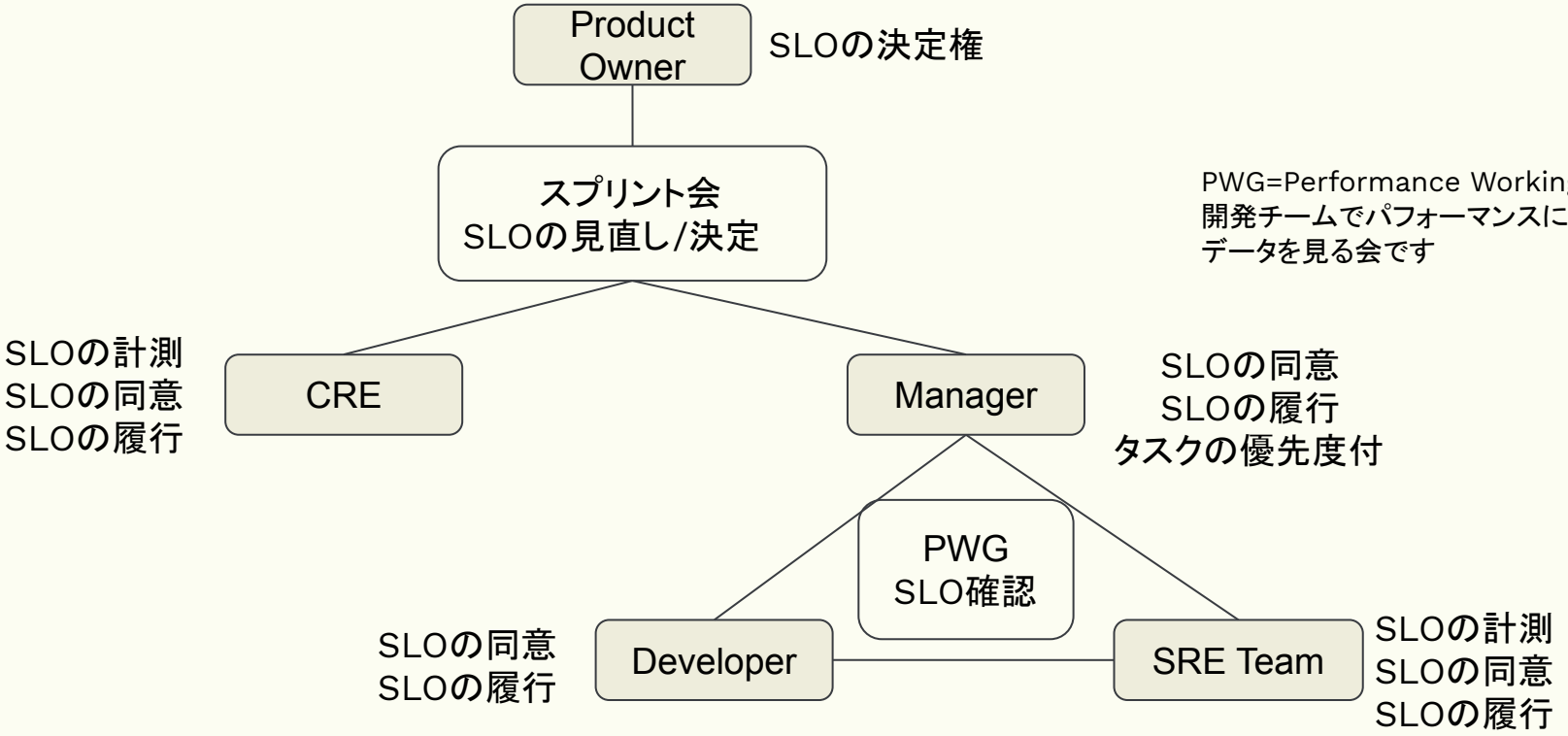


SREと組織 ~ どうやってSLI/SLOを導入するか ~

- (可能であれば)どこかのチームに導入するための準備をする
- サービス開発と同じように構成図やデザインドキュメントなどを書く
  - 例: SLO Docsのレビューや決定をどこでやるのか、誰が何をするのか
- 進め方
  - SLOを設定(SLO Docsを作成)し、SLIを実装する
  - SLIを実装し、SLOを現場から提示する (最初はこちらのほうが楽) ※
    - 最初はモックでも良い

※ ただし、SLOの設定に慣れてきたらSLOから設定したほうがよいと思う (実装しやすいものがSLOになりがち)

(参考) SLOとステークホルダーの図



PWG=Performance Working Group
開発チームでパフォーマンスに関する
データを見る会です

(参考) SLO Docs

Mackerel SLO Document

Mackerelサービス全体の SLO Document です。

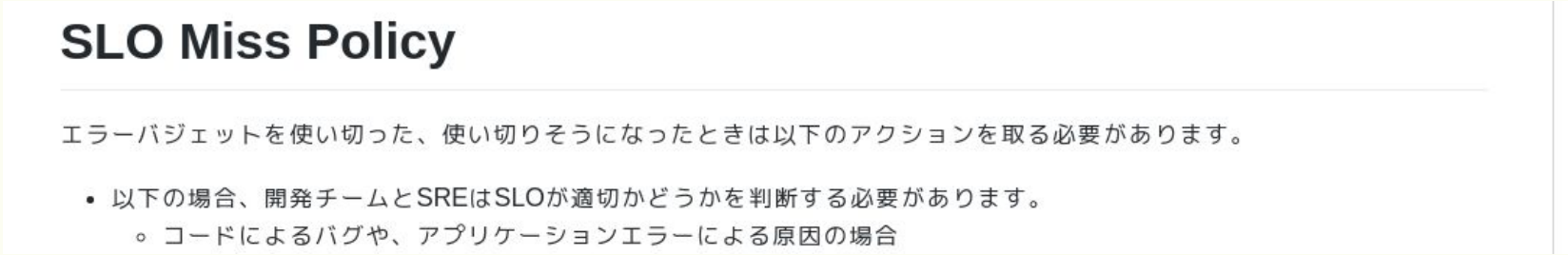
Status	Published
Author	masayoshi
Date	2020-05-08
Reviewers	heleen, hayajo
SRE TL Approvers	masayoshi
Developer TL Approvers	astj
Director Approvers	daiksy
Product Owner Approvers	wtatsuru
Revisit Date	2020-06-05

サービスの概要

- mackerelのAPIサーバと、それらの前段にいるProxy及び、データストアであるRDS(PostgreSQL)と、キャッシュであるRedisで構成される

※ 現在はSLOが適切か判断するために 1ヶ月おきに見直しと更新をしている

(参考) Error budget Policy



SLO Miss Policy

エラーバジェットを使い切った、使い切りそうになったときは以下のアクションを取る必要があります。

- 以下の場合、開発チームとSREはSLOが適切かどうかを判断する必要があります。
  - コードによるバグや、アプリケーションエラーによる原因の場合

※ SLOの調整中なのでリリースの凍結ではなく SLO見直しになるようにしている

SREと組織 ~ その他 ~

- CTOなど決定権をもつ人と共同で進める
- チームをまたいだSREsの横断的組織を作る
  - プラクティスの共有
  - チームをまたいだ調整、組織へのSRE実装、設計
  - SRE成熟度レベルの策定
  - 障害対応ドキュメント, ポストモーテムのテンプレート作成

まとめ

まとめ

- 「これから」はより組織にあったSREを実践、応用していく必要がある
- そのためにはプラクティスを真似るだけではなく、思想、哲学、文化を理解する(考える)ことが重要
- 現段階で発表者が考えているSREの定義やSREの思想、文化の紹介した
- それらの定義、文化とSREを組織に導入していく際の考慮事項をSLI/SLOの導入事例と併せて紹介した

付録

(付録A) SREsを目指す学生、キャリア転向の方向け

発表者のやってきたこと

- 大学、大学院ではSDN(Software Defined Networking)の研究
  - MSP(Managed Service Provider)事業者でサーバ監視のアルバイト
- 2016年 ウェブオペレーションエンジニアとして「はてな」に入社
  - オンプレ環境のネットワーク機器 (Juniper)や仮想化基盤(Xen)の構築、運用
  - オンプレ環境のサーバ、MogileFSストレージサーバなどのクラウド移行
  - CentOS5, MySQL4.0などのレガシーシステムの移行
- 2018年末 ウェブオペレーションエンジニアから SREsに変更
  - 受託案件のサービス開発チームに所属し、リコメンドシステムの構築、運用
  - Mackerel開発チームに所属し、SREテックリードとして、コンテナ化、クラウドネイティブ、SREに挑戦

(付録A) SREsを目指す学生、キャリア転向の方向け

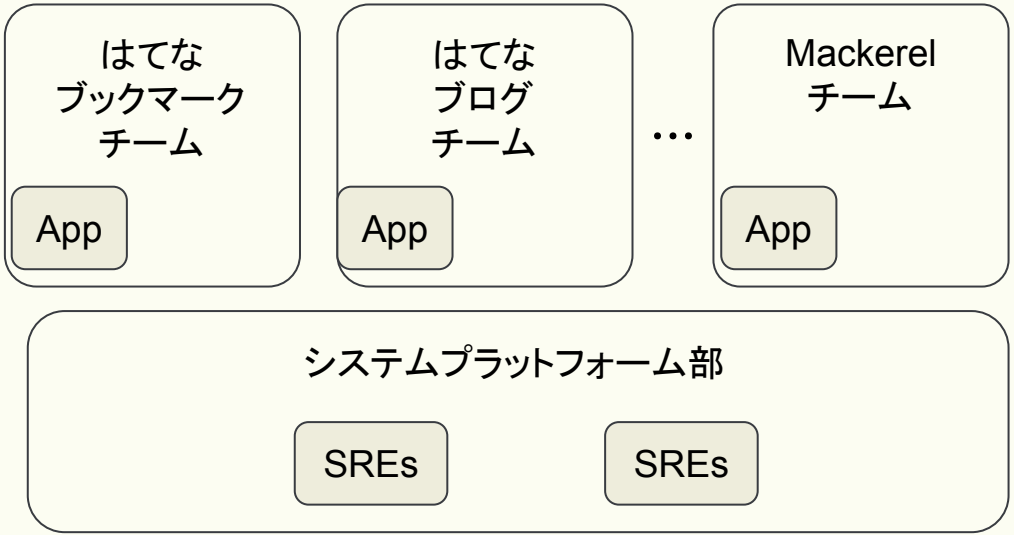
SREsのスキルの話題

- SREを達成するためのスキル = SREs全てに求められるスキル？
  - 今回の発表は「SREに必要なこと」であって「1人のSREsに求められること※」ではない
  - 組織的なこと、チームマネージメントは Manager(EM, PM)が行い、現場のエンジニアがやるなど分業されることも多い
- とはいえ、SREに関わるエンジニアには Soft Skillも重視される傾向がある
  - 採用時の重視するポイント[*1]や SREに求められるスキル項目を見[*2]ても Soft Skillが重視されている
  - 当然技術力が最低限あった上で、Soft Skillも重視されるということ

※ SREsが何(誰)を指すかという定義にもよります
*1 “Defining the Role of a SRE” https://devops.com/defining-the-role-of-a-site-reliability-engineer-sre/
*2 “Training Site Reliability Engineers” https://landing.google.com/sre/resources/practicesandprocesses/training-site-reliability-engineers/

(付録B) SREsと開発チーム

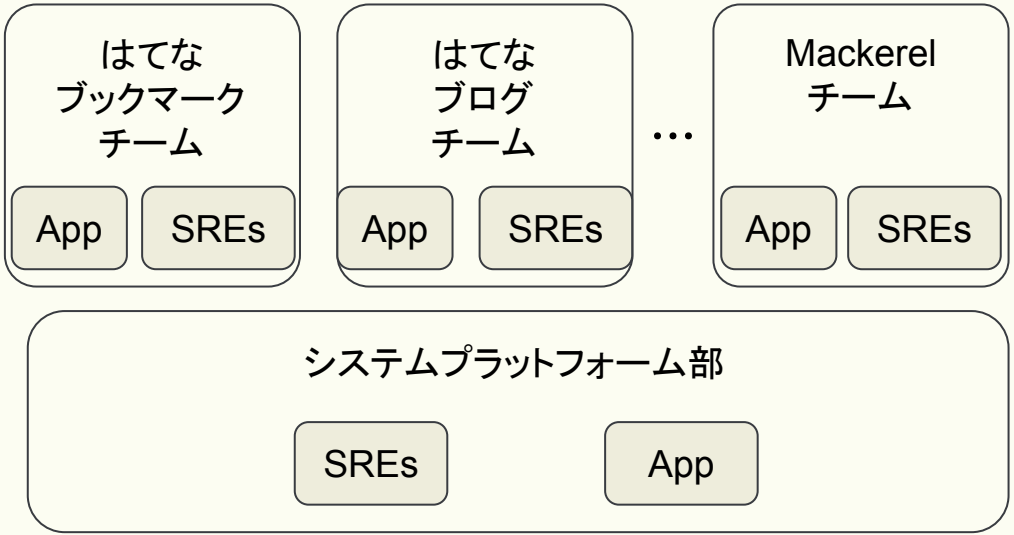
SRE(インフラ)チームの変遷(昔)



※ 当時はSREsではなくウェブオペレーションエンジニアでしたが
わかりやすいようにSREsとしています

(付録B) SREsと開発チーム

SRE(インフラ)チームの変遷(今)



※ 各チームにSREsが一人は配置されるようになりました（徐々に担当者が開発チームへ移籍する）
システムプラットフォーム部にもアプリケーションエンジニアが所属するようになりました

(付録B) SREsと開発チーム

Mackerelチームの場合

- DeveloperとSREsは同じスプリント(開発チームスプリント)
- 「Developerが、SREsがやらないといけない事」はない
  - Dockerfile, Terraform, コードはどちらが書く?などは決めていない
  - 「開発に」、「信頼性に」必要ならやる
  - わからないなら依頼、相談する
  - レビューは「得意な方」に投げる
    - TerraformならSREs, コードの変更はDeveloperなど
- 障害対応はDeveloper, SREs両方やる
  - (現在は)SREsでOnCallを回せるほど人数がいない

(付録B) SREsと開発チーム

Through interviews with different teams, we've found that communication, agreement, and trust are paramount to healthy SRE–developer relationships, and the best functioning teams are those in which it's barely known who is an SRE and who is a developer—who you are is defined by what you do, not your job title.

Training Site Reliability Engineers より

* "Training Site Reliability Engineers" https://landing.google.com/sre/resources/practicesandprocesses/training-site-reliability-engineers/

(付録C) SREと関連項目