



家族アルバム みてねで 直面してきた技術的負債

Vantageスタジオ みてねプロダクト開発部 プラットフォームグループ
清水 勲

SRE観点での技術負債 懺悔会 2024



清水 勲 @isaoshimizu

家族アルバム みてね Engineering Manager (SRE/CRE/Security)

新卒入社

ミクシィ (現MIXI) 入社

2022年1月～EM

Sler時代 (受託・自社開発)

SNS 「mixi」

モンスタースト
ライクなど

みてね

2003年

2011年

2014年

2018年

2024年

C/C++/C#/PHP/Python/iOS/AWS

Fedora/MySQL/LXC
/OpenStack

Linux/MySQL/Ruby

AWS/MySQL/Ruby

週末は社会人吹奏楽団での活動 (楽団長、トロンボーン約30年、たまに指揮者)。

キャンプとクラフトビールが好き。

New Relic User Group 運営

技術的負債

“Technical liabilities that cost us because they create obligations – obligation which slow us down, obligations which must be repaid; and they create risks, we don’t know when an obligation is going to bite us.”

“技術的な負債は、私たちを遅らせる義務、返済しなければならない義務、
そしていつその義務が私たちに襲いかかるかわからないリスクを生み出す。”

Allan Kelly: Technical liabilities: not technical debt (2016.12.4)
<https://www.allankelly.net/archives/415/technical-liabilities-not-technical-debt/>

技術的負債による影響は . . .

- 開発者体験が悪くなる
- 品質が悪くなる
 - 機能開発がやりづらくなる
 - リリースが遅れる
 - ユーザー体験が悪化する

技術的負債と

どうやってうまく向き合っていくか

- 優先順位の問題
 - プロダクトの価値やユーザー体験の向上を犠牲にしたくないこととの葛藤
- 開発しながら日々返済する
 - ときには一気に返済する必要も（ビジネスへの影響が大きい）
- 過去の経緯を理解しながら向き合う
 - とはいえ経緯がわからないこともあるかもしれない（つらい）
 - 負債を生むことを前提に設計、判断をしているわけではないことを理解する

家族アルバム みてねのSRE視点で 直面してきた技術的負債

(数年前から振り返ってみました)

Fluentdにおける様々なトラブル

解決済み

- だいぶ昔の話です（6年くらい前）
- 長い間 各EC2インスタンスごとのForwarder -> Aggregator(複数台) -> S3という構成だった
- ユーザー数も増えトラフィックも増えていく中でFluentdが不安定に
- Fluentd自体のアップデートもあまりされていなかった
- **開発者がFluentdの障害対応にかかりきりに（機能開発への影響も）**
- 各種パラメータ調整して試行錯誤するもうまく安定化せず
- SREがFluentd -> Kinesis Data Firehose -> S3という構成に変更して安定化
- 現在はFluent Bit(DaemonSet) -> Data Firehose -> S3

AWS OpsWorksの様々な課題

解決済み

- EKSに乗り換える前の話（2021年以前）
- オートスケールが遅い（1台追加されるまでに5分とか普通）
- スポットインスタンスが使えない
- WebUIでポチポチ作業
- Chefレシピのメンテナンス
- 各種パッケージのアップデート漏れ（オートスケール時EC2が使いまわされる）
- EKSに乗り換えてすべて解決
- (参考) 4年間のEKS移行の取り組みを振り返って

<https://gihyo.jp/article/2022/11/mitene-02eks>



郵便番号データ（ken_all.zip）への依存

要改善

- <https://www.post.japanpost.jp/zipcode/download.html> からダウンロードできる日本全国の郵便番号データ
- 以前は不定期に手動で更新していた（更新用のコマンドがあった）
- 一時的に新しい住所に対応できていなかったことも（ごめんなさい）
- いまではCronJobによって定期的に自動更新
- しかしCSVをパースするPerlスクリプトが残ったまま
 - このためだけにPerlが必要なのはいまいち...
- 郵便局のサイトに依存している
- ケンオールに移行できると色々解決しそうだがまだやれていない

巨大なDBテーブルのスキーマ変更

要改善

- サービスの成長に伴って増え続けるDBレコード
- 一部のテーブルはとにかくデカい（数百億レコードあるものも...）
- pt-online-schema-changeでは対応できないレベルに（つらい）
- gh-ost を試してみたい
 - <https://github.com/github/gh-ost>
 - 試すにはAuroraのバイナリログが必要
 - 最近のAuroraでは拡張バイナリログによって性能劣化は最小限に

Amazon Linuxアップデート

解決済み

- Amazon Linux 1 -> 2 -> 2023 という変遷
 - つい最近までAmazon Linux 2を使い続けていた
 - NodeのEOL問題（つらい）
- OSをアップデートする際には、利用しているツールやミドルウェアのアップデートも一緒にやらなければならない
- 公式パッケージ構成が大きく変わって苦勞すること
- 詳しくは「レガシーなシステムに向き合うということ」を参照
 - <https://team-blog.mitene.us/a201fff59b42>



FFmpeg / ImageMagickのアップデート

一部要改善

- 画像や動画を扱うサービスとして欠かせないソフトウェア
- 依存ライブラリのアップデートで挙動がよく変わる（つらい）
- メジャーアップデートで挙動がよく変わる（つらい）
- ということもあって積極的にアップデートがしづらく、バージョンが古いままになりやすい
- とはいえ脆弱性対応や新機能の利用などでアップデートは必要
- アップデート時の挙動が変わったことに気付けるようにテストを充実させておくことが大事
- FFmpeg 絶賛アップデート対応中（ImageMagickは最新に！）

まとめ

- 扱うサービス、ソフトウェアが増えるほど技術的負債が増える可能性
- 便利だからといって採用したライブラリがある日負債になるかもしれません
- 技術的負債は日々少しずつ返していきましょう（一度にドカッと返すのは大変）
- 放置される期間が長いほど返済が大変に（成長が激しいサービスほど大変）
- この数年で多くの負債を解消してきましたが、まだまだあります
- ユーザーのためにも、開発者のためにも技術的負債と向き合っていきます

心もつなごう。

MIXI

