JAPAN | JUNE 20, 2024

aws SUMMIT

D O L - 2

AWS CDKコンストラクトの実例から見る開発組織効率化とライブラリ開発の妙

友岡 雅志

Prototyping Engineer

内容

AWS CDKコンストラクトの実例から見る、開発組織効率化とライブラリ開発の妙

- 想定聴講者
  - 開発組織の効率化やライブラリ開発に興味のある開発者の方
- 話すこと
  - ライブラリ開発に関する知見を筆者の実体験を交えて紹介します

- 話さ**ない**こと
  - CDKライブラリ開発の始め方
  - CDKに特化した知見

妙の解説 - 小学館 デジタル大辞泉

みょう【妙】

［常用漢字］［音］**ミョウ**（メウ）（呉） ビョウ（ベウ）（漢） ［訓］たえ

1 言うに言われぬほど美しい。「妙音／美妙」

2 奥深く味がある。きわめて巧みである。「妙案・妙手・妙味・妙薬／軽妙・玄妙・巧妙・神妙 (しんみょう)・神妙 (しんびょう)・精妙・絶妙・即妙・微妙・霊妙」

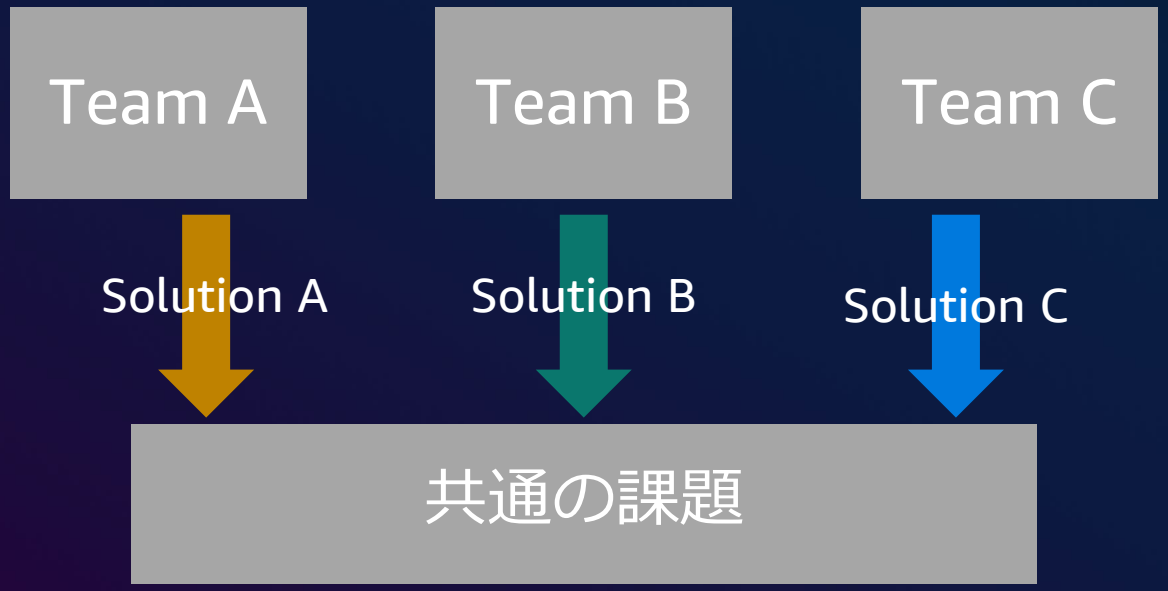
3 うら若い。「妙齢」

4 変わっている。不思議だ。「奇妙・珍妙」

［名のり］ただ・たゆ

Motivation: 開発プロセスを効率化したい！

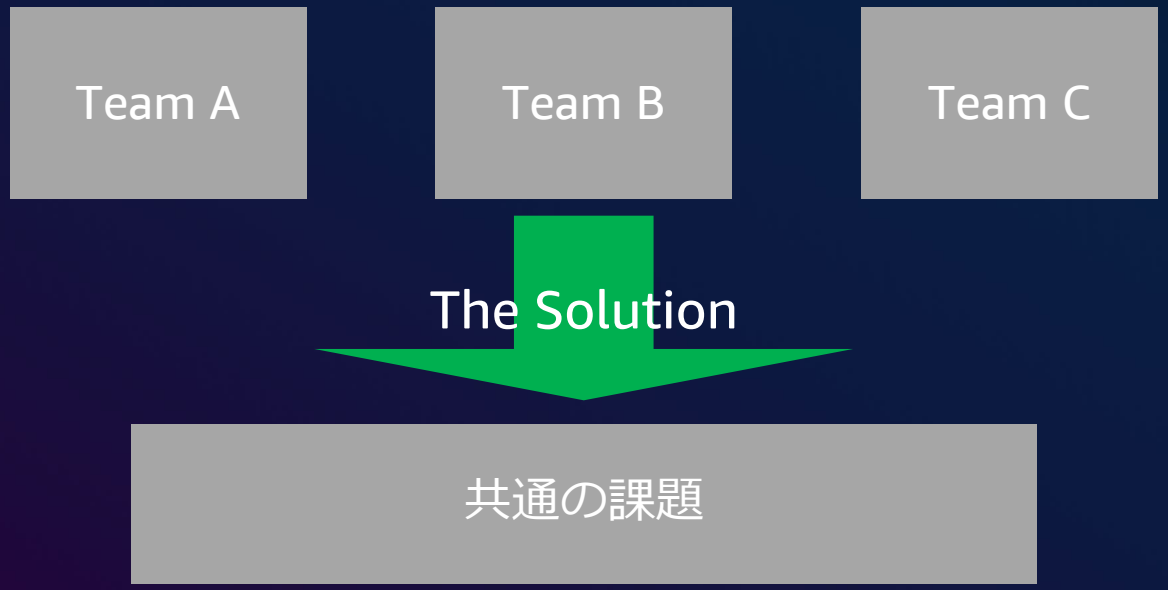
個々のチームがそれぞれで解決策を編み出す状況



“車輪の再発明”

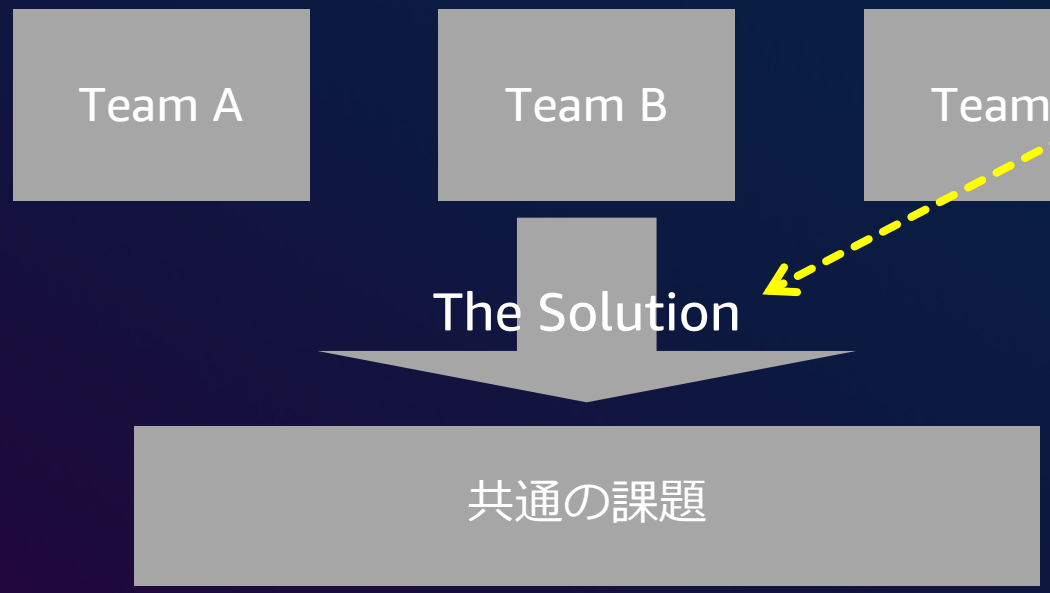
Motivation: 開発プロセスを効率化したい！

一つの改善案: 共通の課題に対して**共通のソリューション**を用意する



Motivation: 開発プロセスを効率化したい！

一つの改善案: 共通の課題に対して**共通のソリューション**を用意する



現実は困難が伴う

- リリース後、あまり利用されなかった
- そもそもどの課題を解決すべきか不明
- ユーザー増の結果、要件が複雑化

Motivation: 開発プロセスを効率化したい！

一つの改善案: 共通の課題に対して**共通のソリューション**を用意する

現実は困難が伴う　今日はこちらにフォーカス！

- リリース後、あまり利用されなかった
- そもそもどの課題を解決すべきか不明

  → 初期フェーズの課題

- ユーザー増の結果、要件が複雑化

  → 拡大フェーズの課題

自己紹介

友岡 雅志 **プロトタイピングエンジニア** @AWS Japan

前職

- mBaaS開発・運用 (Rails, Sinatra, MySQL)
- モバイルゲームクライアント開発 (Unity, C#)

現職

- お客様のプロトタイプ開発を支援
- こちらもチェック！ プロトタイピングのススメ (AWS Summit 2023)

AWS CDK関連ライブラリを開発しています Construct Hub

- 最近週間ダウンロード数合計が5,000を超えてきた

𝕏: @tmokmss

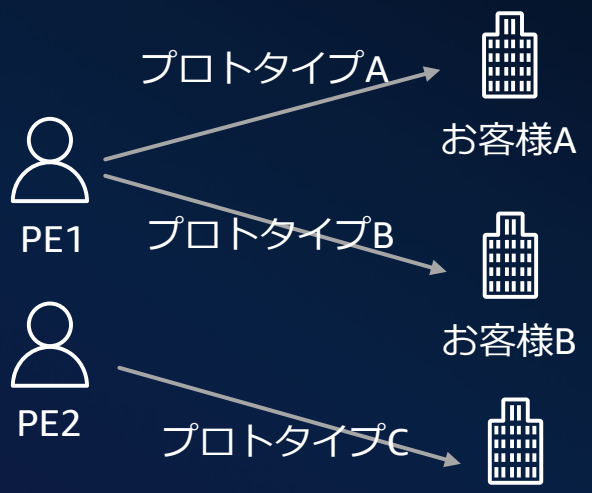
AWSプロトタイピングチームと開発効率化

Prototyping Program

ビジネスニーズにあわせたプロトタイプを開発することで、
お客様のイノベーションを加速する

参考:
プログラムの概要

- 約1件/月/人のペースでプロトタイプを開発
- → 車輪の再発明は発生しがち
- → デリバリー効率や業務体験向上の余地あり



PE = Prototyping Engineer

一つの手段: AWS CDK の活用

AWS Cloud Development Kit (CDK) とは

- 汎用言語 (TypeScriptなど) で記述する Infrastructure as Code ツール
- 我々プロトエンジニアにとっては「まな板」のような不可欠の仕事道具
- 「**コンストラクト**プログラミングモデル」を提唱

CDK コンストラクトの魅力

- IaCコードの複雑性を内部に隠蔽できる (かつ柔軟性は損なわれない)
- 再利用可能な**ライブラリ**として配布可

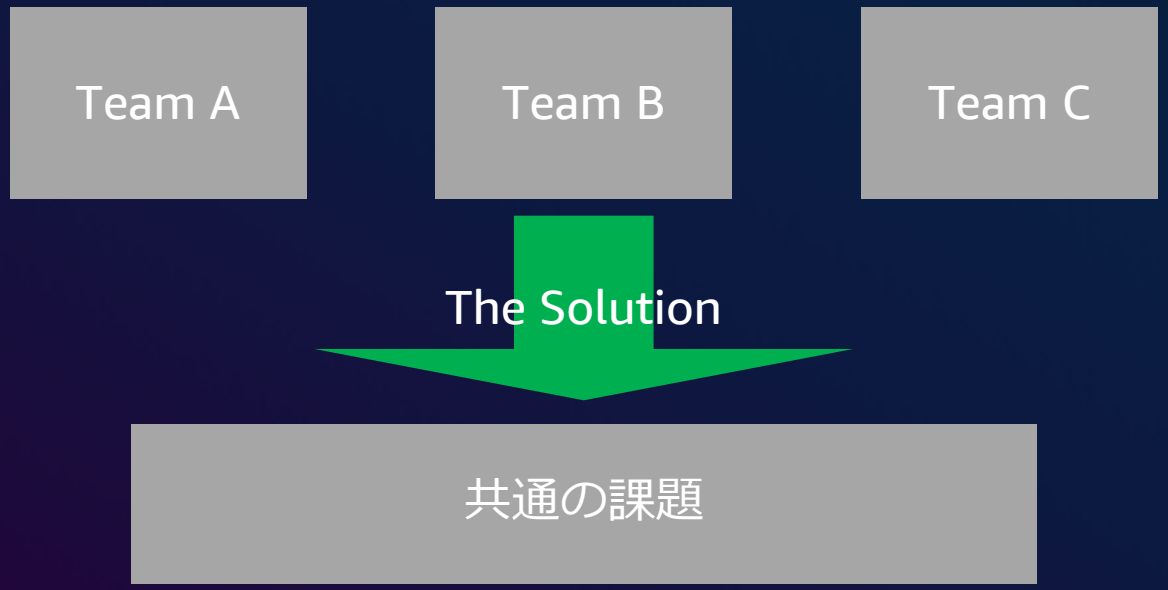
本日はCDK利用方法のうち、特にライブラリ開発の側面にフォーカスし、汎用的な知見を導きます

例: 踏み台サーバーを定義するCDKコンストラクト

```
const host = new BastionHostLinux(this, 'Bastion', {
  vpc,
});
```

Motivation: 開発プロセスを効率化したい！(再掲)

共通の課題に対して共通のソリューションを用意することによる効率化



現実は困難が伴う　今日はこちらにフォーカス！

- リリース後、あまり利用されなかった
- 解決すべき課題を見誤った

  → 初期フェーズの課題

- ユーザー増の結果、要件が複雑化

  → 拡大フェーズの課題

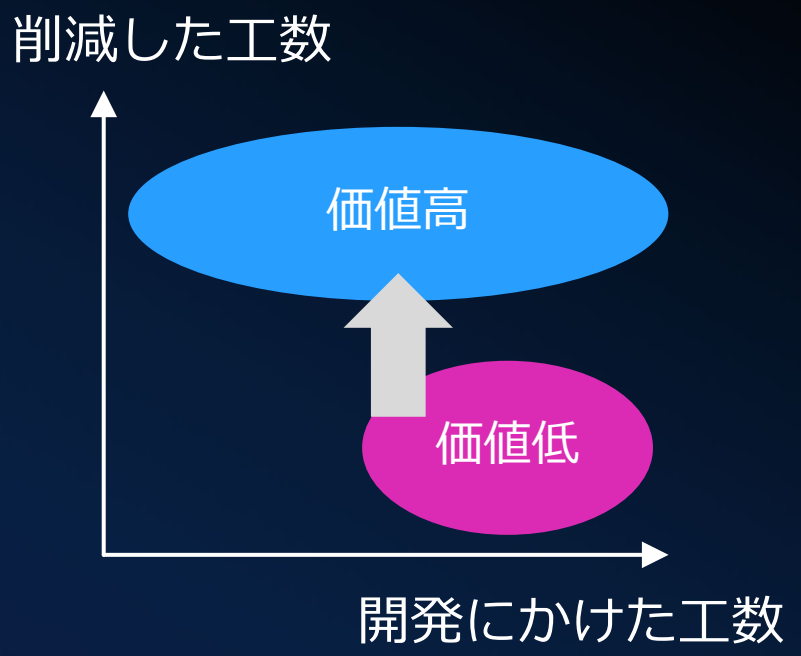
本日のテーマ: ライブラリ開発の初期フェーズを いかに「成功」させるか

ライブラリ開発における**成功**とは？

→ ライブラリが価値を発揮している状態

→ 価値 = **削減した工数 ÷ 開発にかけた工数**

→ 削減した工数は**ユーザー数**に比例する



成功にはどれだけ
ユーザーを獲得できるかが鍵

(潜在的)ユーザーたちの思考を想像する

思考の分類

ライブラリ開発者として考えたいこと

1. **ライブラリの存在意義**: 何をなぜ作るのか
2. **UX設計**:ライブラリのユーザー体験を考える
3. **広報活動**: どうやって存在を知ってもらうか

ライブラリの存在意義 = 何をなぜ作るか

- コツ: 需要ベースで考える
  - **潜在的なユーザー数を探る**: 社内やインターネット上に困っている人がいるか？
  - **既存の代替手段を探す**: 何をどう解決しているか それらの改善できる部分は何か
    - もし代替手段が存在しないなら、それはなぜか？

- そもそも解決策 = ライブラリの開発**とはならない**問題もある
  - あまりにも単純な問題は、スニペットの提供で十分かもしれない
  - 解決策を共通化できるほど問題を絞り込めない
    - → 将来の複雑化が容易に予見できる

実際の例 – 自作CDKコンストラクトライブラリより

- 事例1: deploy-time-build
  - S3+CloudFrontへのSPAの静的ファイルへ環境変数を埋め込む方法を提供
  - 上記の方法についてはチーム内で**度々議論の的**となっていた
  - 議論を通じて既存手法のPros/Consは整理されていたため、**差別化は容易**だった

- 事例2: opensearch-rest-resources
  - OpenSearchのRESTリソース (FGACなど) のCloudFormation管理を可能に
  - re:PostやGitHub Issue, 社内Slackなどで**同じ課題感**を持つ開発者が散見された
  - 当時、同じ課題に対する代替の解決策は**存在しない**状況だった (理由は...)

潜在ユーザーを見出す・ライブラリは妥当な解法か検討

ライブラリ開発者として考えたいこと

1. **ライブラリの存在意義**: 何をなぜ作るのか
2. **UX設計**: ライブラリのユーザー体験を考える
3. **広報活動**: どうやって存在を知ってもらうか

UX指向のAPI設計

- **API**: (ここでは) どのようにライブラリを呼び出すかの決め事
  - 例: 関数の名前、引数の与え方、戻り値の受け取り方など
    ライブラリにおいてはユーザー体験 (UX) への影響が大きい
- コツ: **ユーザー体験を起点**に考える
  - まずはUXを追求してAPIを設計する　実装の詳細を考えるのはその後
    - APIを実装の詳細に引きずらせない　不要なことをユーザーに意識させない
  - 使いやすさの要因
    - **単純さ**：より単純なコードで目的を達成できれば、使いやすい
    - **慣れ**：従来の方法と似たような使い方ができれば、使いやすい

実際の例

- 事例1: deploy-time-build
  - 入力: フロントエンドのソースコード
  - 出力: SPAをビルドした静的ファイル
  - 目的: 環境変数を静的ファイルに埋め込む

CDKユーザーには**慣れ親しんだ**
Assetインターフェースや
BucketDeploymentの作法を流用

問題を解決するために**最低限の情報だけ**
入力させる - 不要なことは意識させない

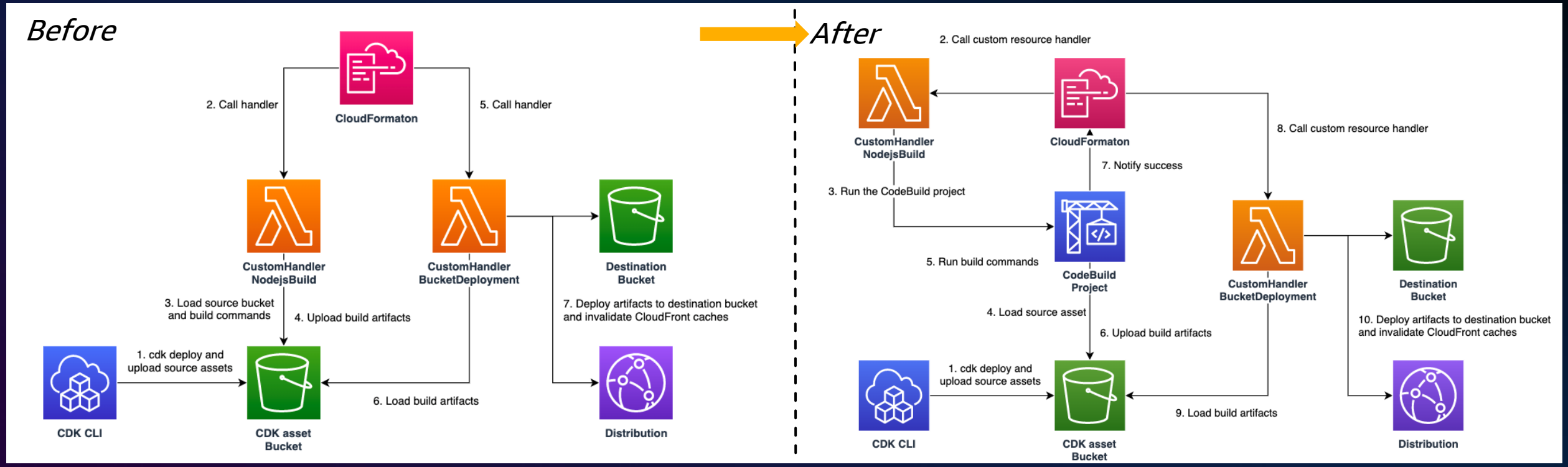
```
import { NodejsBuild } from 'deploy-time-build';

declare const api: apigateway.RestApi;
declare const destinationBucket: s3.IBucket;
declare const distribution: cloudfront.IDistribution;
new NodejsBuild(this, 'ExampleBuild', {
    assets: [
        {
            path: 'example-app',
            exclude: ['dist', 'node_modules'],
        },
    ],
    destinationBucket,
    distribution,
    outputSourceDirectory: 'dist',
    buildCommands: ['npm ci', 'npm run build'],
    buildEnvironment: {
        VITE_API_ENDPOINT: api.url,
    },
});
```

例(続き): 実装の詳細 (implementation detail) を隠す

内部改善のため実装の詳細を変更したが、APIはそのままで済んだ

deploy-time-build のアーキテクチャ図　(新旧)



慎重なAPI設計は多くの利益をもたらす

ライブラリ開発者として考えたいこと

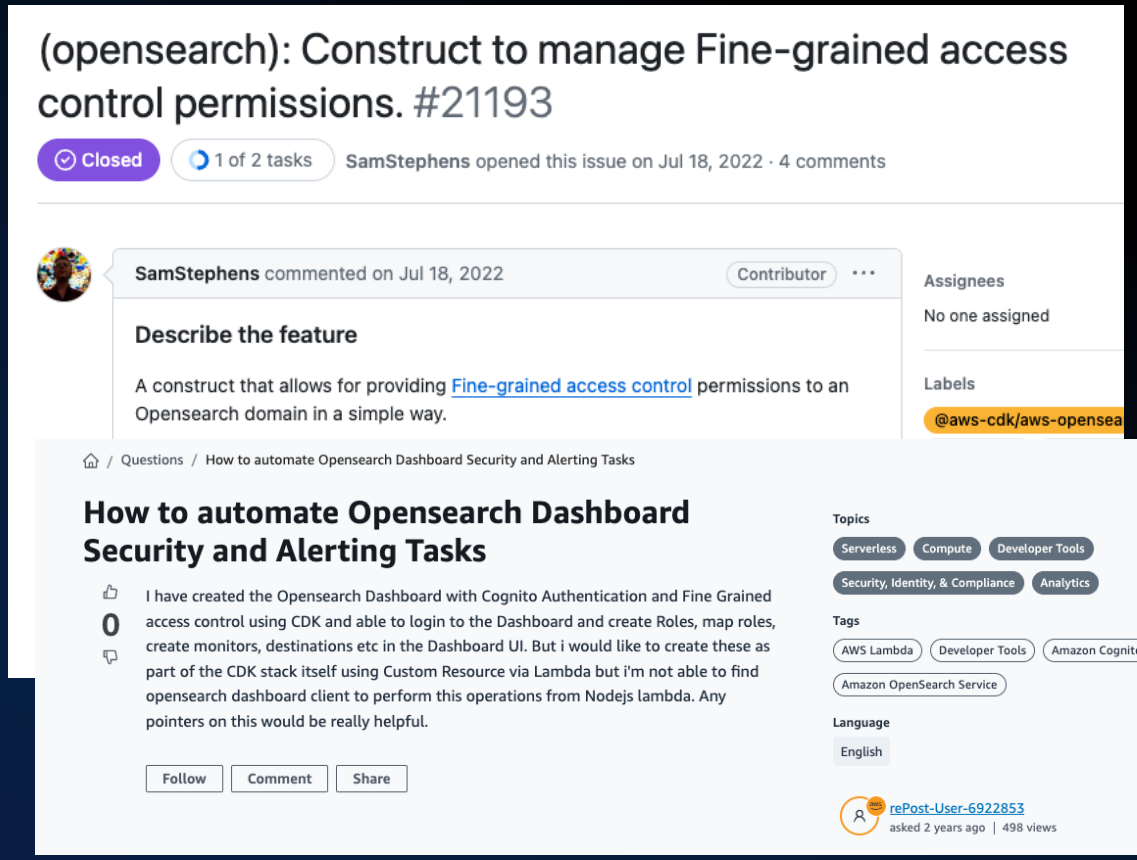
1. **ライブラリの存在意義**: 何をなぜ作るのか
2. **UX設計**:ライブラリのユーザー体験を考える
3. **広報活動**: どうやって存在を知ってもらうか

広報活動: 利用者の増やし方

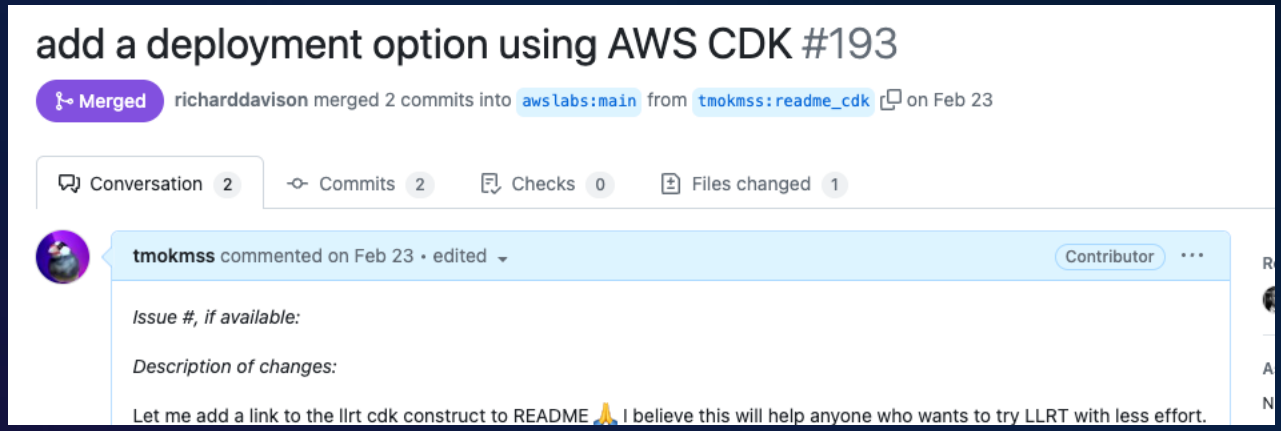
- 前提: ユーザーは存在を知らないライブラリを採用することはない
  - ユーザーに存在を知らしめるため、効果的な宣伝が重要
- コツ
  - 特定のユーザーへ**直接宣伝**に行く
    - ライブラリの存在意義を決めたときに、潜在ユーザーは見出だせているはず
  - 検索流入 (SEO) の意識
    - **困っているユーザー**が検索しそうなワードで検索ヒットするように
    - GitHub IssueやStackoverflowは検索にインデックスされやすい
    - ブログや登壇資料も◯ (ユーザー間で**紹介しやすく**なる)

実際の例

- 事例1: opensearch-rest-resources
  - 同僚に共有 → 確実な利用者を獲得
  - 検索でヒットするGH Issueやre:Postに解決策を投稿した → 1000 downloads/week到達



- 事例2: cdk-lambda-llrt
  - LLRTのLambda関数を作るライブラリ
  - 𝕏でLLRTのバズりに便乗
  - LLRT公式のREADMEにも掲載を依頼
  - → 1500 downloads/week到達



ソフトウェアライブラリの好循環 / Flywheel

隠れたバグが見出される＝”枯れる”
優良な追加機能が提案される

ユーザーの増加

共通課題の解決

ライブラリ採用のメリットが
デメリットを上回る

ライブラリ開発者として考えたいこと

1. **ライブラリの存在意義**: 何をなぜ作るのか
2. **UX設計**:ライブラリのユーザー体験を考える
3. **広報活動**: どうやって存在を知ってもらうか

まとめ

- 開発効率化のために、**ライブラリ開発**が効果的な手段になり得る
- ライブラリ開発の初期フェーズを成功させるために考えたいこと
  - ライブラリの**存在意義**
  - **UX設計**
  - **広報活動**

もちろん、これらがすべてではありません！

ぜひライブラリ開発の妙の世界に飛び込み
新しい学びを得てみましょう！

参考資料

- AWS CDKに興味を持った方へ: AWS CDK Workshop
- CDKのAPI設計に対する考え方も参考に: AWS Construct Library Design Guidelines
- 先人・偉人たちからも学ぶ (非OSSな開発でも転用できる知見はあります)
  - OSSで世界と戦うために
  - Re: OSSで世界と戦うために
  - 個人開発OSSが世界に勝てなかった話

AWS CDK Conference Japan 来月開催！

2024/7/6 12:00-18:30
@目黒 (+オンライン)

Connpass

私も話します

今こそ始める、CDKコンストラクトライブラリ開発 — 入門から実践まで

AWS Summit Japan 2024 盛り上がっています！

私的おすすめセッション

- **DOL-4 フィードバックドリブンに生成AI活用アセットを開発した話**
  - 同じくプロトタイピングチームのメンバーよりお届けします
  - OSS開発観点では、拡大フェーズの知見も含まれる

- **AWS-33 チームのつながりを Infrastructure as Code でデザインする**
  - IaC (CDK) を起点に考える組織・プラットフォーム改善の話

Thank you!

友岡 雅志

mtomooka@amazon.co.jp

GitHub / 𝕏: @tmokmss