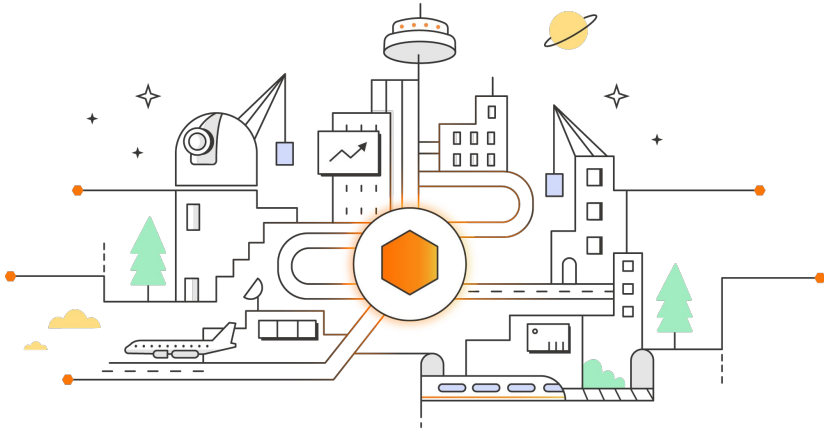
POSTMAN

APIを腐らせないために

長く使われるAPIを支える運用の現実とAI時代の運用論

Cloud Operator Days Tokyo 2026

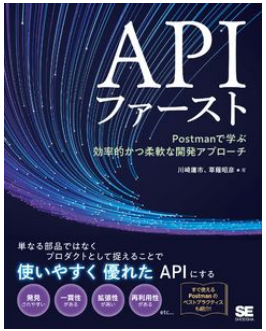
川崎庸市
テクノロジーエバンジェリスト



川崎 庸市 / Yoichi Kawasaki

Postman株式会社にて、テクノロジーエバンジェリストとして同社製品の導入支援、社外登壇、執筆活動を通じて、同社製品ならびにAPI技術の啓蒙・教育に取り組む。加えて、SaaSスタートアップの技術顧問も務める。以前は、国内EC事業会社や外資系ソフトウェアベンダーにて幅広いエンジニアリング業務に従事。

@yokawasa



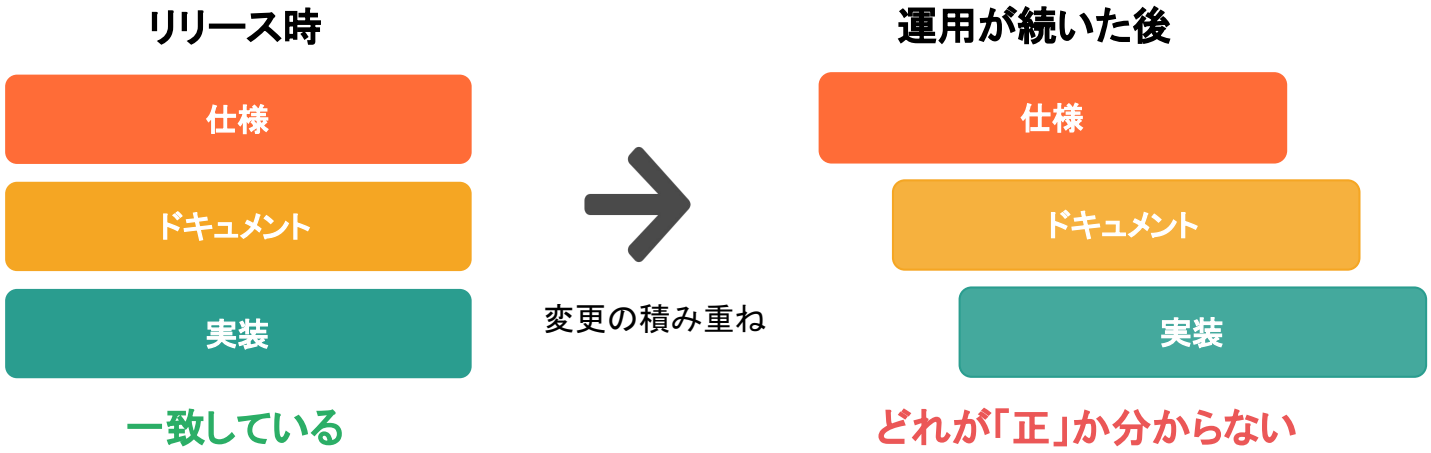
話すこと

- API運用の現場で実際に起っている問題
- AIがAPIの開発者・利用者になったら？
- APIを腐らせない（=資産にする）ために、私たちは何をすべきか
- まとめ

API運用の現場で実際に起っている問題

APIドリフト（実装と記述とのズレ）

- 最初は一致していた仕様・ドキュメント・実装が、変更のたびにじわじわ乖離していく
- やがて、どれが「正」なのかが誰にも分からなくなり、運用が推測ゲーム化する



APIドリフト（実装と記述とのズレ）

公開されているAPIの約30%が、公開されているAPI記述と一致していない

The challenges of API Drift - most production APIs don't match their specs (Aug 2024, APIContext社)
https://apicontext.com/resources/api-drift-white-paper/

API可視性の欠如 → 車輪の再発明

「どこに何のAPIがあるか」が分からず、すでに在るものが何度も作り直される

可視性がない

- API一覧がどこにもまとまっていない
- 検索で見つからない／存在を知らない
- 担当者の頭の中だけにある

車輪の再発明

- 同じ機能が別チームで再実装される
- 微妙に違う重複仕様のAPIが乱立
- 保守対象が増え、悪循環となる

一貫性のないAPIの増殖

標準化されていない、あるいは標準化が守られておらず、一貫性のないAPIができあがる

	サービスA	サービスB	サービスC
命名規則	`/getUser`	`/users/{id}`	`/user_info`
エラー形式	`{ error: "..." }`	`{ code, message }`	`HTTP 200 + 本文にerr`
認証方式	`API Key (header)`	`Bearer Token`	`Query param ?key=`
日付表現	`UNIX time`	`ISO 8601`	`"YYYY/MM/DD"`

シャドーAPIの増殖

- シャドーAPI = 管理台帳を通さず、組織に把握されないまま稼働しているAPI
- 単なる技術的負債に留まらず、組織を揺るがす深淵なるセキュリティ問題を生む可能性あり

把握されていない

API

いつのまにか本番に

なぜ生まれるか？

- 短納期で「とりあえず」公開される
- 検証用・社内用が外部に流出する
- 正式なAPI登録・リリースの仕組みがない

何が怖いか？

- セキュリティ・監査の対象から漏れる
- 障害時に存在自体に気づけない
- サイバー攻撃の格好の裏口になる

事例：T-MobileのAPI侵害

2023年、T-Mobileは約3,700万人の顧客情報がAPI攻撃により漏洩したと発表しました。

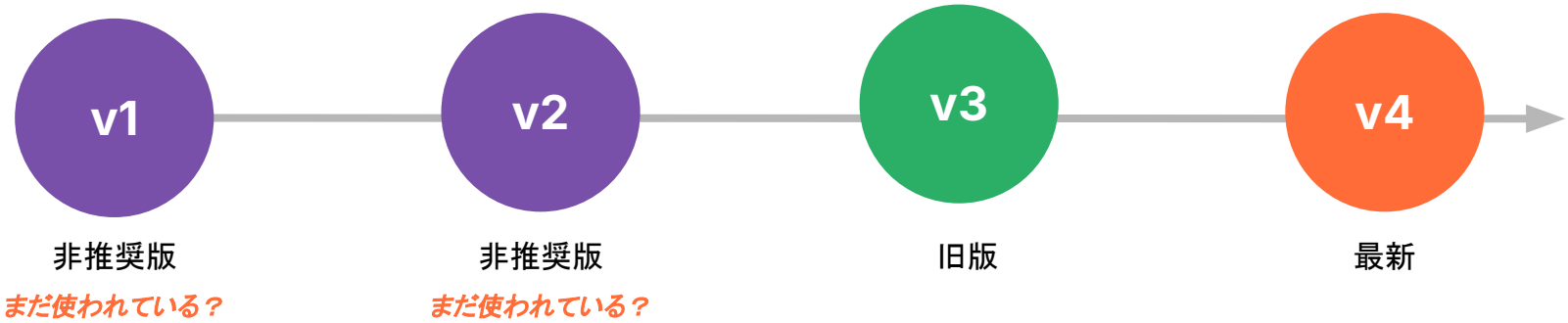
攻撃者は、**認証不備のあるシャドー API**を悪用し、不正アクセスを継続していました。

この事件は、未把握のAPIが重大な脆弱性になることを示す深刻な事例です。

出典: T-Mobile API Breach — What Went Wrong?
https://salt.security/blog/t-mobile-api-breach-what-went-wrong

ゾンビAPIの増殖

- ゾンビAPI = 非推奨扱いのAPIにも関わらず、いつまでも使われ続けるAPI
- いづれ監視・メンテもされなくなり、セキュリティインシデントの突破口になる



なぜ終わらせられないのか？

- 誰が使っているか把握できていない
- 止めると何が壊れるか読めない
- 念のために残し続け、バージョンだけが積み重なる

まとめ - API運用の現場で起っている問題

- APIドリフト（実装と記述とのズレ）
- API可視性の欠如 → 車輪の再発明
- 一貫性のないAPIの増殖
- シャドーAPI、ゾンビAPIの増殖
- 合意形成と廃止プロセスの不在

これらの問題が、担当者の暗黙知や気合いによって何とか支えられてきた現場は少なくない

AIがAPIの開発者・利用者になったら？

AIが利用者になる

明文化されていない仕様は機能しない

人間なら空気を読んで暗黙の前提を補完していたが、AIは補完しない

これまで - 利用者が人

- 曖昧でも文脈で察する
- 詰まれば人に聞く
- 慣習や行間を補って使う

曖昧さ・暗黙知を人が補完

これから - 利用者がAI

- 書かれた仕様の通りにしか動けない
- 暗黙の前提は機能しない
- 曖昧さは即エラー・誤呼び出しに

暗黙知が通用しない

AIが開発者になる

生成のスピードと量が、桁違いになる

- 人間の管理・認知能力を超えた速度でAPIが生まれる
- 結果、シャドーAPIが量産され、似て非なる仕様の不整合が増殖する可能性が高まる

AIによる生成量とリスク

1. 量の爆発（将来予測）
エンタープライズアプリの**40%** が、2026年末までにタスク特化型AIエージェントを組み込む（2025年比8倍超）
出典：Gartnerプレスリリース

2. 量の実測
AIツールを使う開発者は、平均**3〜4倍** のコミットを生成

3. 品質リスク
同じツールが **10倍** のセキュリティリスクを生む
出典：Apiiro (2025.9)

量が桁違いに増え、品質もばらつく。人手のレビューは限界へ

管理が追いつかない

登録・レビューが生成速度に置き去りにされる

不整合の増殖

似て非なる仕様が大量に生成される

シャドーAPIの量産

把握されないAPIが指数的に増える

ガバナンスの変化

人手によるガバナンスが限界を迎える

人が確認して止めるという、人手によるガバナンスが効かなくなる

これまで - 人間中心

- 人手のレビューで品質を担保
- 暗黙の合意・口頭での調整
- 人が読む前提のドキュメント

人手によるガバナンス

これから - AI中心

- 速度・量にレビューが追いつかない
- 明文化されない仕様は機能しない
- 機械可読な形式での提供が必須になる

人手によるガバナンスが効かない

これまで人間が吸収していた問題が一気に増幅される

曖昧さ

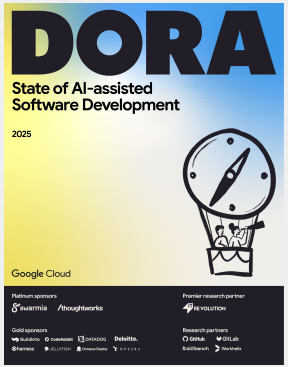
AIはエラー・誤動作。
書かれた情報だけが頼り

速度と量

生成が管理速度を超え、
シャドーAPIが量産

ガバナンス

人が見て止める前提が効かない



AIは増幅器 (an amplifier) である

DORA 2025 State of AI-assisted Software Development Report

APIを腐らせない（=資産にする）ために
私たちは何をすべきか

人の暗黙知や管理能力に頼らない仕組みの導入

5つの打ち手

SSoT

SSoTで一元化する

発見可能

APIを発見可能にする

規律

APIの登録・廃止プロセスに規律を設ける

計測

利用者と変更の影響を捉える

ガードレール

ガバナンスと品質のガードレールを敷く

打ち手

SSoTで一元化する

一次情報とする単一情報源（SSoT）を定め、SSoTによる一元化を行う

1 SSoTの設定とSSoTによる一元化

- SSoTとそれを元に作られる派生物の関係を明確化する
- SSoTからの派生物の生成はできる限り自動化する

2 機械可読な仕様を作る

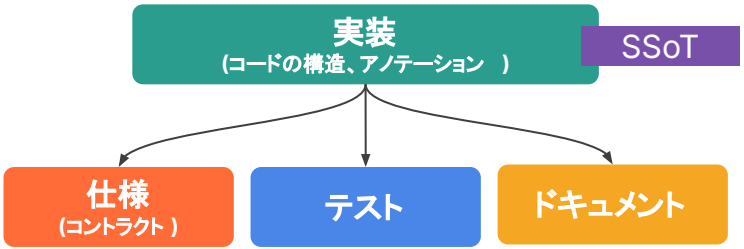
- 仕様の機械可読化することで、色んな形で自動化に活用できる
- 仕様からドキュメント、SDK、モック、テストの生成など

＊ SSoT = Single Source of Truth

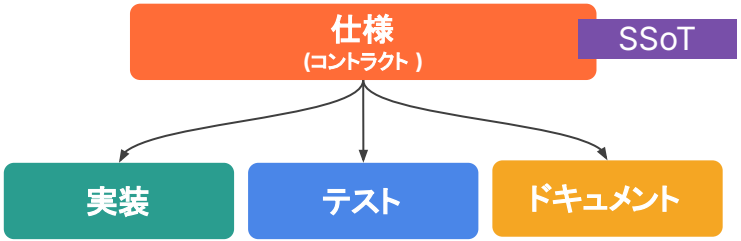
SSoTによる一元化の例

一次情報 / 単一情報源（SSoT）を設定し、派生物の生成をできるかぎり自動化する

コードファーストの例



デザインファーストの例



機械可読なAPI仕様の例

OpenAPI 3.1.0

```yaml
paths:
  /users/{id}:
    get:
      summary: ユーザーを1件取得
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: string
      responses:
        '200':
          description: 成功
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:   { type: string }
                  name: { type: string }
        '404':
          description: 該当ユーザーなし
```

Postman Collection 2.1

```json
{
  "info": { "name": "User API" },
  "item": [
    {
      "name": "ユーザーを1件取得",
      "request": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "https://api.example.com/users/:id",
          "host": ["api", "example", "com"],
          "path": ["users", ":id"],
          "variable": [
            { "key": "id", "value": "123" }
          ]
        }
      }
    }
  ]
}
```

打ち手

APIを発見可能にする

いま何があるかをカタログ化し、車輪の再発明を止める

1

カタログ化を行い、APIを作る前に探せる状態にする

- APIを一覧化し、横断検索できる状態にする
- 作る前に既存のAPIを見つけられる状態を作り、APIの重複を防ぐ

2

カタログは機械可読な形式でも提供し、AIからも発見・利用可能にする

- 人ばかりでなくAIにとっても発見できる状態にする
- 例: APIやMCP経由で取得可能なAPI一覧情報

打ち手

APIの登録・廃止プロセスに規律を設ける

- APIの登録・廃止方法に規律を設け、シャドーAPIやゾンビAPIの発生を防止する
- APIの登録・廃止管理プロセスをAPI設計に含める

1

API登録プロセスに規律を設ける（シャード API防止対策）

- 未登録APIは公開できない仕組みを作る
- APIをOpenAPI登録必須にして一元化したり、公開時はゲートウェイへ登録必須とするなど

2

API廃止プロセスに規律を設ける（ゾンビ API防止対策）

- 廃止に関する詳細情報（対処 API、バージョン、停止日、移行先など）の用意
- 機械可読な通知（ Deprecationヘッダー、など）と、ブログ・メールなど人向けの両方を用意

機械可読なAPI廃止の通知 - HTTPヘッダー例

廃止予定APIのレスポンス例

```
HTTP/1.1 200 OK
Content-Type: application/json
Deprecation: @1782873600
Sunset: Tue, 31 Dec 2026 23:59:59 GMT
Link: <https://docs.example.com/api/migrations/users-v1-to-v2>; rel="deprecation"
Link: <https://status.example.com/api-sunsets/users-v1>; rel="sunset"
Link: <https://api.example.com/v2/users/123>; rel="successor-version"

{"id": "123","name": "Yoichi Kawasaki","email": "yoichi.kawasaki@example.com"}
```

- Deprecation : このAPIは非推奨になった、または非推奨になることを伝えるヘッダー。RFC 9745で定義
- Sunset : このAPIが将来使えなくなる見込みの日時を伝えるヘッダー。RFC 8594で定義
- Link : 関連情報へのリンク。rel="deprecation" は廃止理由や移行方法の説明へ、rel="sunset" は停止予定や停止ポリシーへ、rel="successor-version" は後継APIへ誘導する意図で使う。RFC 9745で定義

RFC 9745 https://datatracker.ietf.org/doc/rfc9745/
RFC 8594 https://datatracker.ietf.org/doc/html/rfc8594

打ち手

利用者と変更の影響を捉える

API利用者の現状を把握をして、コントラクトとバージョニングにより破壊的変更から守る

1 APIの利用状況を計測する（現状把握）

- 誰がどのAPI・どのバージョンを利用しているかを把握する
- 廃止判断やゾンビ API対策に直結

2 コントラクトとバージョニングで守る（予防）

- 期待する振る舞いをコントラクト（契約）として明文化し、破壊的変更を検知する
- 避けられない変更はバージョンを分け、利用者の移行猶予を確保する

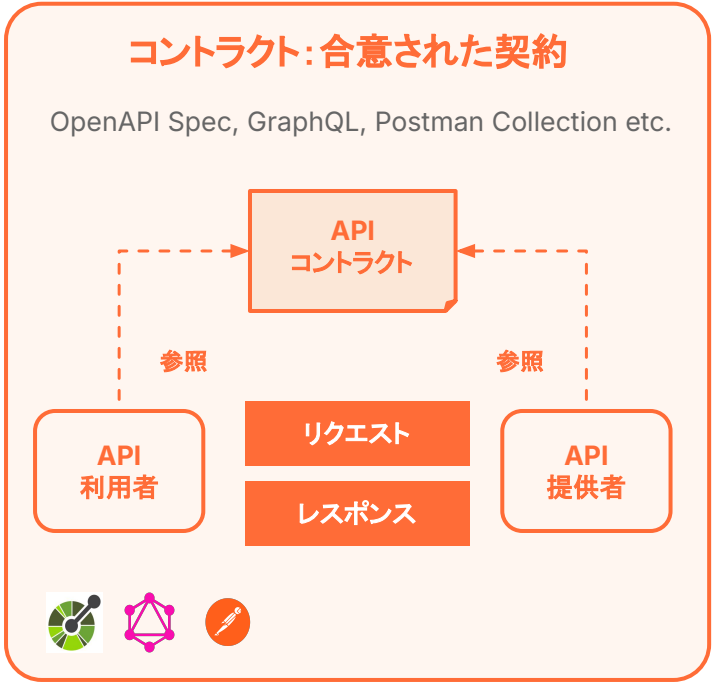
コントラクトテスト

概要

- APIが合意された「コントラクト」通りに振る舞うか検証するテスト
- 提供者はリグレッション、利用者は変更検知として定期実行を推奨

主なチェック箇所例

- リクエスト・レスポンスのスキーマ整合性
- 型（Type）と値（Value）
- 必須パラメータ、フィールドの有無
- エンドポイントとHTTPメソッドの動作
- 適切なHTTPステータスコードの返却



打ち手

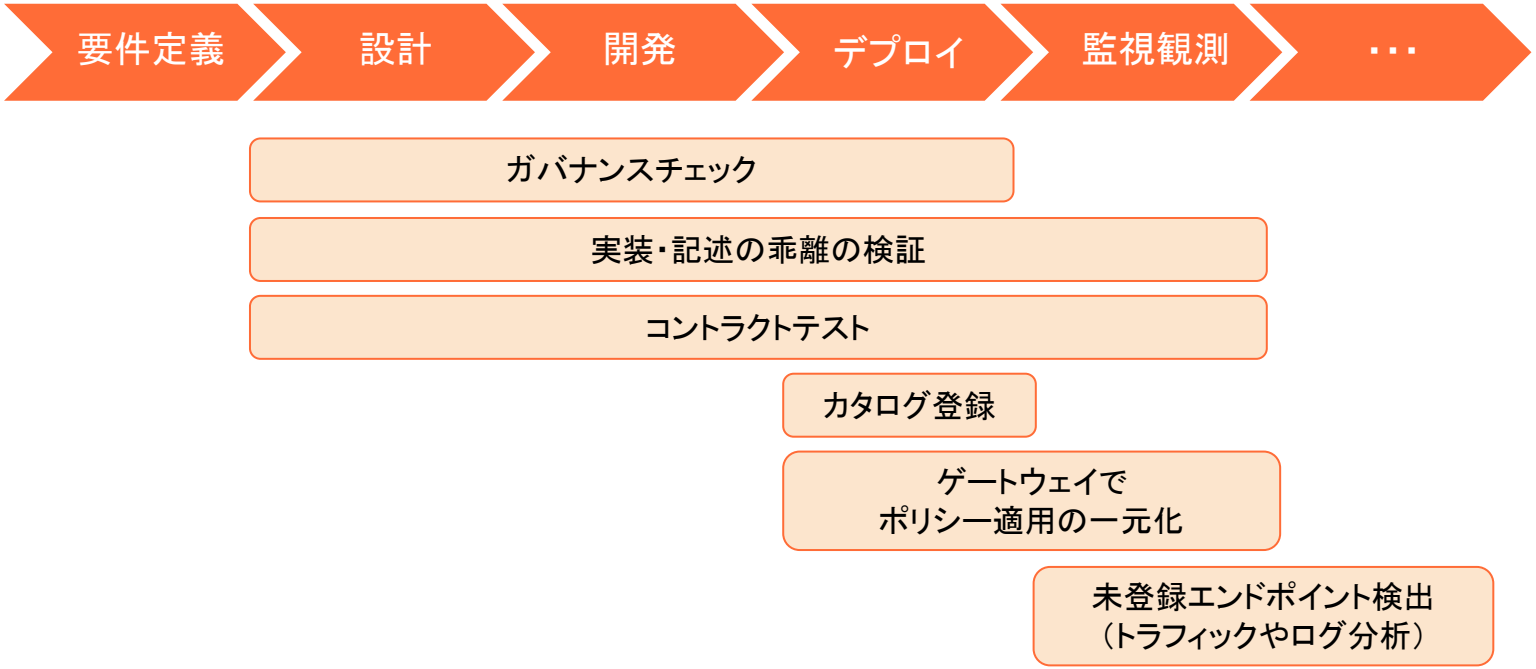
ガバナンスと品質のガードレールを敷く

- 人手のレビューはスケールしない
- 特に高速にAPIが生まれるAI時代、ガードレールがスケールするための防衛線になる

打ち手	対応するガードレール例
SSoTで一元化	• ガイドライン準拠・ガバナンスを自動チェック（Linting） • 実装・記述の乖離の自動検証
発見可能	• 公開にはカタログ登録を必須化する関門を設ける
規律（登録・廃止）	• 実装・記述の乖離の自動検証（未登録APIを弾く） • API通信のポリシー適用を中央制御点（APIゲートウェイなど）で一元化 • 実トラフィックやログ分析で未登録エンドポイントの検出
計測（利用者・変更影響）	• コントラクトテストで破壊的変更を検知
その他	• ユニット・シナリオテスト・E2E（機能面） • セキュリティ・パフォーマンステスト（非機能面）

シフトレフト - 運用課題への対処を前倒しする

問題は早期発見・早期対処が最も効率的。後工程で対処するほどコストは大きくなる。



Postman - 統合APIプラットフォーム

- APIライフサイクルの実行、コラボレーション、ガバナンス、ガードレールを支援
- 組織全体でAPIの発見・再利用が可能となり、安全性と一貫したガバナンスの実現が可能

提供する機能・能力

- APIクライアント
- 設計 & モック
- テスト自動化
- ドキュメンテーション
- モニタリング & オブザーバビリティ
- APIカタログ
- エンタープライズ コントロール
- セキュリティ & ガバナンス

実現できること

- APIライフサイクルの標準化
- APIの可視化
- APIスプロール (乱立)の抑制
- ガバナンスのシフトレフト
- ガードレール

Postman - 統合APIプラットフォーム

長く使われるAPI運用のための5つの打ち手を支援

Postman (APIプラットフォーム)

仕様をSSoTとした派生物の自動生成	APIカタログによる可視化	APIライフサイクルの標準化	コントラクトテスト	強力なガバナンス、テスト実行基盤
SSoT	**発見可能**	**規律**	**計測**	**ガードレール**
SSoTで一元化する	APIを発見可能にする	APIの登録・廃止プロセスに規律を設ける	利用者と変更の影響を捉える	ガバナンスと品質のガードレールを敷く

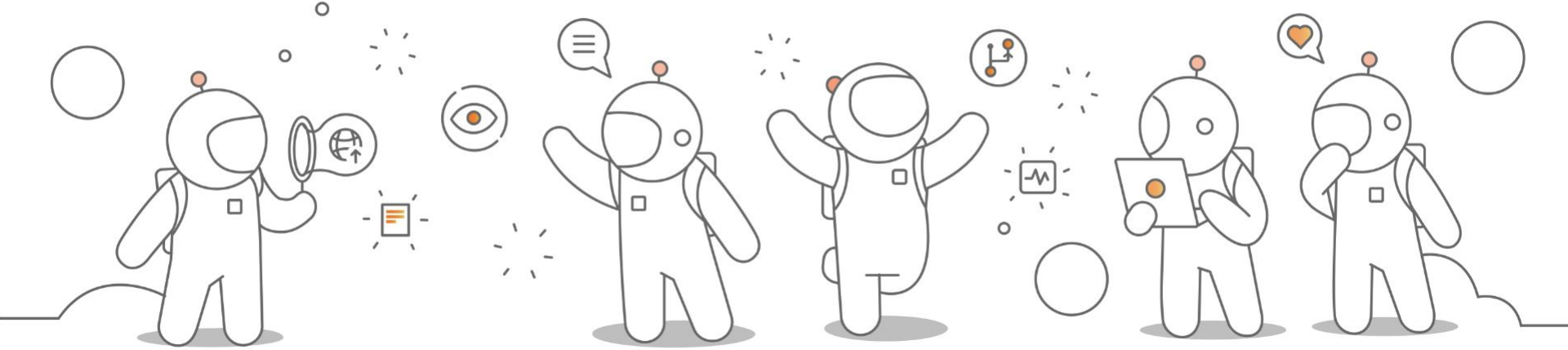
まとめ

まとめ

APIを腐らせない（=資産にする）ために、私たちは何をすべきか

- APIの運用現場には、古くから根深い問題がある
  - 仕様と実装の乖離、シャドー /ゾンビAPI、合意形成・廃止プロセスの不在など、これまで人間が暗黙知と気合いで何とか回してきた
- AIが開発者になる時代、これまで人間が吸収していた問題が増幅される
  - 曖昧さはエラーになり、生成の速度と量は人の管理能力を超え、人が見て止めるガバナンスは効かなくなる
- 人の暗黙知や管理能力に頼らない仕組みの導入
  - 5つの打ち手：SSoTで一元化する、発見可能にする、登録・廃止に規律を設ける、利用者と変更の影響を捉える、ガバナンスと品質のガードレールを敷く

ありがとうございました



@postman_japan