API入門と企業がAPIを公開するプロセス

～API公開事例から学ぶ～

株式会社オージス総研

サービス事業本部 クラウドインテグレーションサービス部

齋藤 伸也（Saito_Shinya@ogis-ri.co.jp）

目次

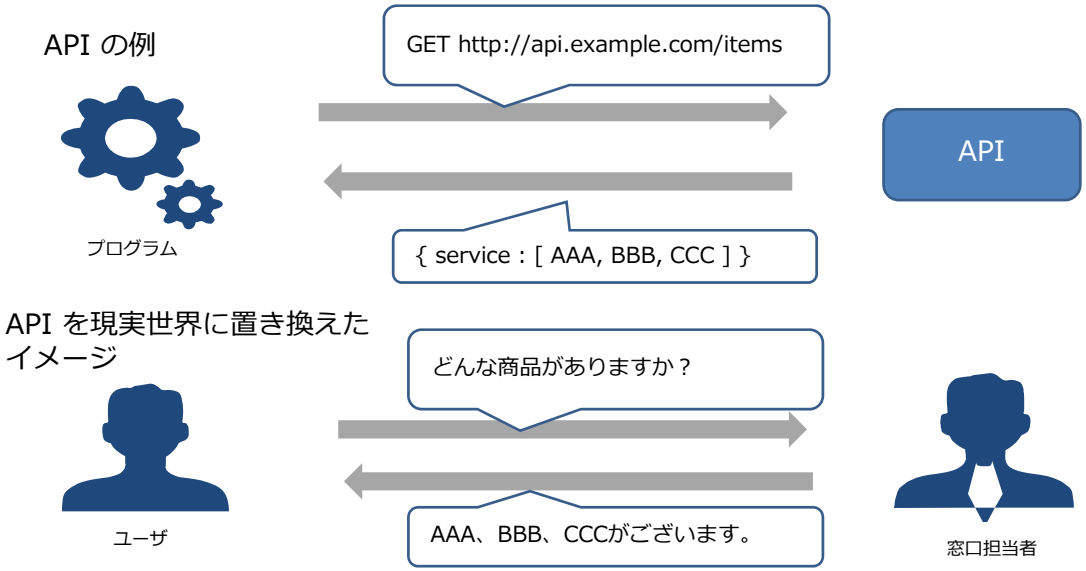
- API入門
- 事例概要
- 事例におけるAPI公開プロセス
- APIソリューションご紹介

オージス総研

API入門

API とは?

- 最近注目されている”API”は企業が持つデータやサービスを、他のアプリケーションやプログラムから利用するための窓口を意味する
- “API”はHTTPなどのWebの技術を用いて構築されたプログラムから利用可能なインタフェース



APIの技術的背景

- プログラミングやソフトウェアの相互運用性を確保するための技術や仕様の策定が行われる
  - 1998年～2003年：XML-RPC、SOAP、WSDLなどの仕様が策定される
  - 2000年：RESTが提唱される
  - 2000年代後半：GoogleやAmazonなど大手Webサービス企業がAPIの公開を始める。当時はSOAP、REST両方のスタイルでAPIが提供されていた。現在はRESTのみの提供。
- 最近のAPIはデータ形式としてJSON 、RESTスタイルが採用されることが多い

データ形式: XMLとJSON

XML: 表現力が豊か、厳密性

```
<?xml encoding='utf-8' ?>
<user>
  <name>saito</name><age>32</age>
  <name>yamada</name><age>25</age>
  <name>kimura</name><age>41</age>
</user>
```

JSON: シンプル、相互運用性

```
{"user" : [
  { "name" : "saito", "age" : "32" },
  { "name" : "yamada"  , "age" : "25" },
  { "name" : "kimura" , "age" : "41" }
]}
```

スタイル: SOAPとREST

SOAP: 基本的な考えはリモート関数呼出。URIは関数の集合を表す名詞

POST http://domain/api/itemSearchService 商品一覧取得
POST http://domain/api/itemRegisterService 商品登録

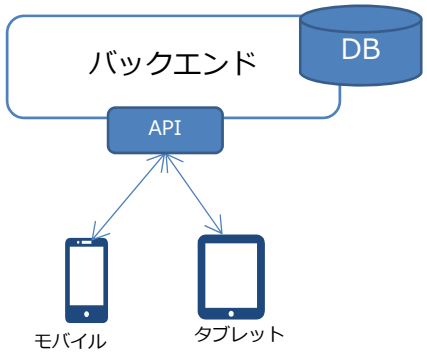
REST: 基本的な考えはHTTPの原理。URIはリソースを表す名詞

GET http://domain/api/items 商品一覧取得
POST http://domain/api/items 商品登録

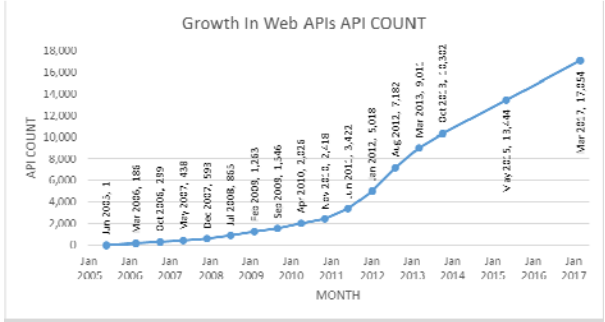
APIの普及

- 2003年頃～ Amazon、Googleなど大手Web企業がAPIを提供開始。Ajaxの普及
- 2007年頃～ AWS, Salesforce, Twitter, Facebook等クラウドサービスがAPIを提供開始
- 2009年頃～ スマートフォンの普及、モバイルアプリの開発が活発化。モバイルアプリのサーバ(バックエンド)とデータをやり取りする。仕組みとしてAPIの普及が本格化
- 2014年頃～ IoT、フィンテックやAPIエコノミーが注目される非IT企業APIへの取組み本格化

モバイルアプリとAPI



増加を続けるAPI

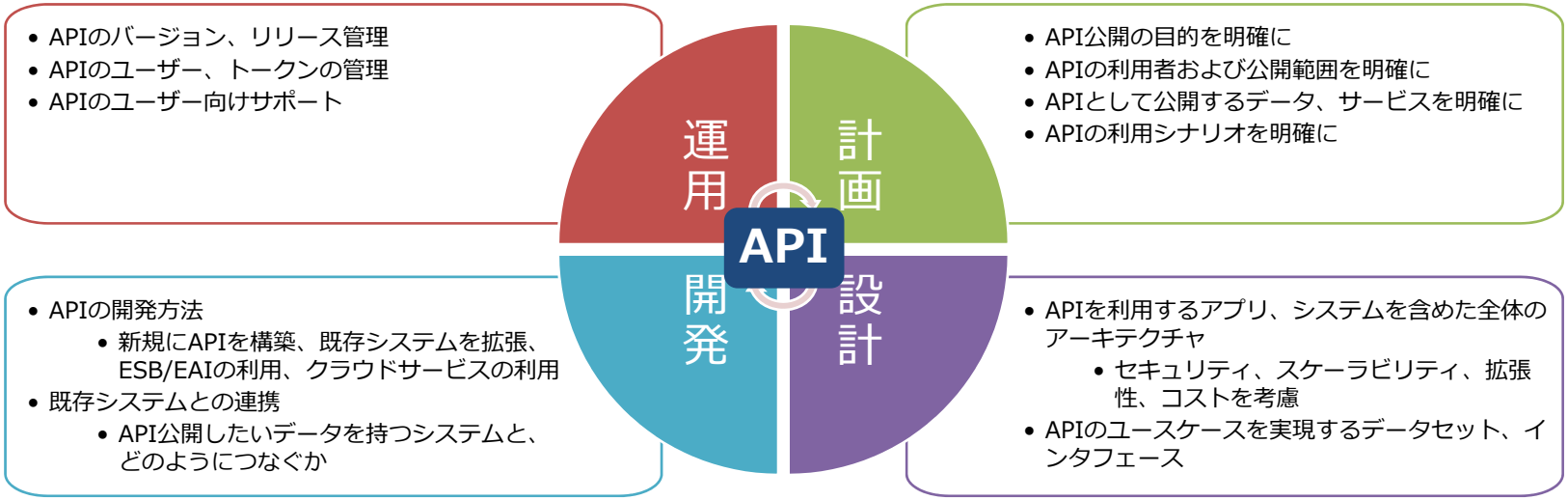


ProgrammableWebの情報を基に当社で加筆・グラフ化
引用元: https://www.programmableweb.com/api-research

API公開のライフサイクル

- API公開は、一度きりの取り組みではない
- デジタルビジネスの成長、変化にあわせAPIを改修し、バージョンアップすることが必要

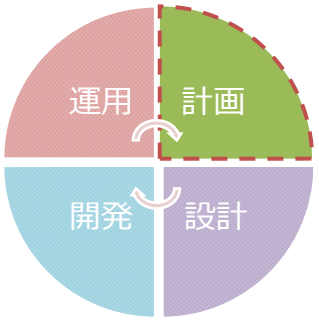
  → ライフサイクル管理が大切



API公開のライフサイクル:計画

API公開の計画で重要になるポイント

- API公開の目的を明確にする
- APIの利用者および公開範囲を明確にする
- APIとして公開するデータ、サービスを明確にする
- APIの利用シナリオを明確にする



API公開範囲の種類について

プライベート

メリット： 様々なクライアントから共通的に利用可能なモジュールを提供できる。

例: モバイル向けのバックエンドAPI

パートナー

メリット: パートナーとの新規協業、立ち上げの迅速化ができる。

例: 取引先、代理店向けのカタログAPI

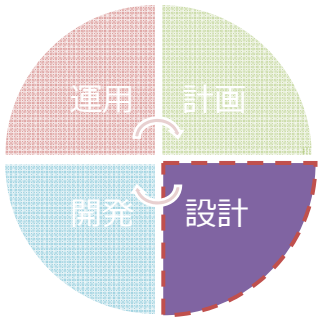
パブリック

メリット: ビジネスをプラットフォーム化することを実現できる。

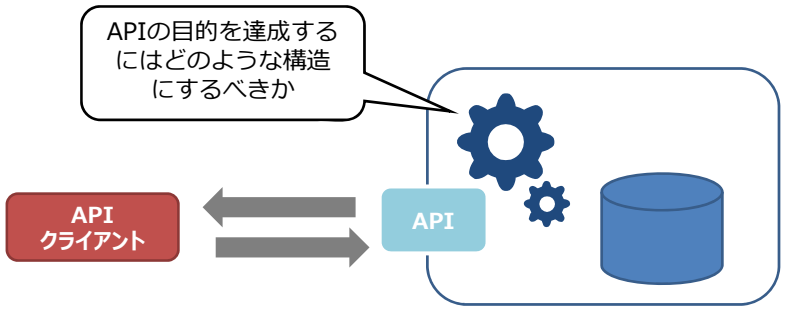
例: オープンに公開されているMap API

API公開のライフサイクル:設計

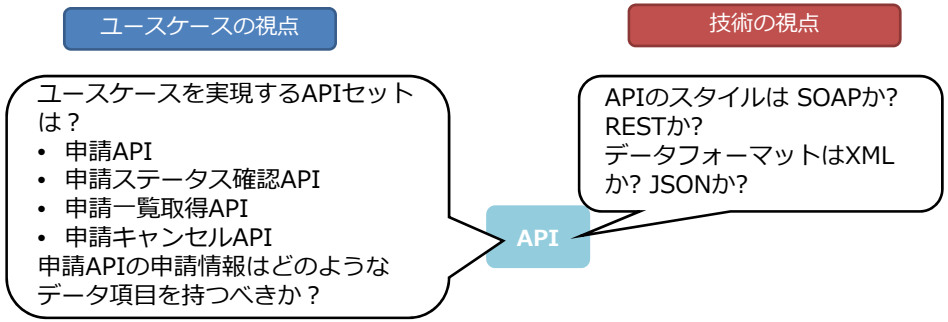
- API公開の設計で重要になるポイント
  - APIを利用するアプリ、システムを含めた全体のアーキテクチャ
    → セキュリティ、スケーラビリティ、拡張性、コストを考慮する
  - APIのユースケースを実現するデータセット、インタフェース
    → ユーザ視点のデータセット、標準的なAPIスタイルなどユーザの利用しやすさを考慮する



アーキテクチャ設計



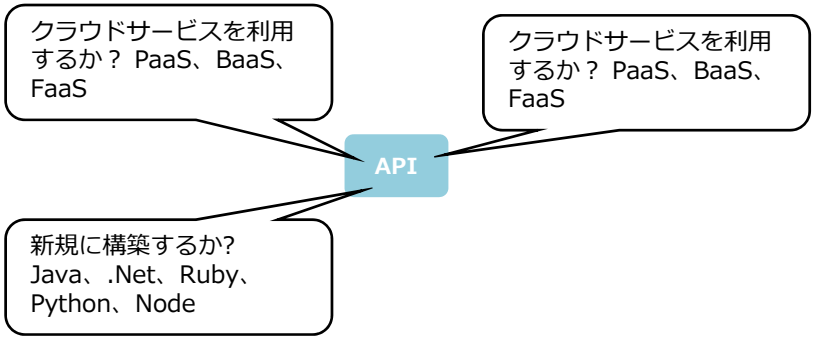
インタフェース設計



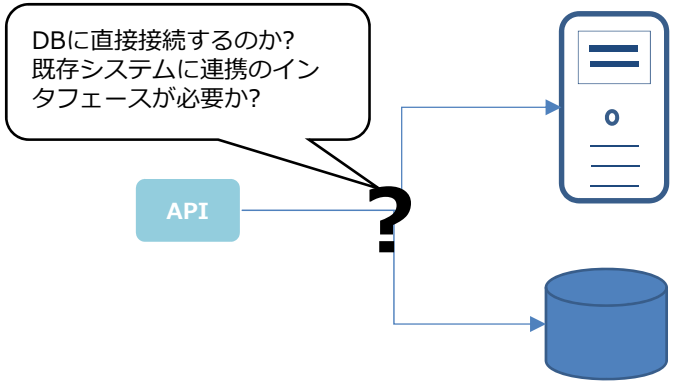
API公開のライフサイクル:開発

- API公開の開発で重要になるポイント
  - APIの開発方法
    - → 新規にAPIを構築、既存システムを拡張、ESB/EAIなどの連携ミドルウェアの利用、クラウドサービスの利用
  - 既存システムとの連携
    - → API公開したいデータを持つシステムと、どのようにつなぐか

何をつかって開発する？

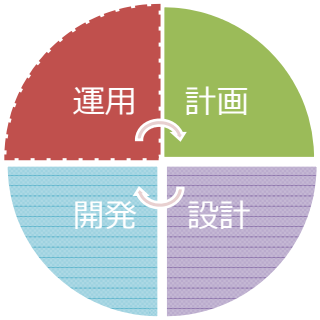


どうやって連携する？



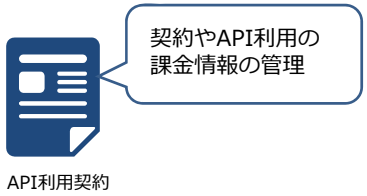
API公開のライフサイクル:運用

- API公開の運用で重要になるポイント
  - APIのバージョン、リリース管理
  - APIのユーザ、契約管理
  - APIのユーザ向けサポート
  - APIの監視、障害対応



APIの機能追加やデータ項目
変更などの管理する

	バージョン	ライフサイクル
ユーザプロファイル 変更API	1.2	公開中
サービス取得API	0.1	開発中

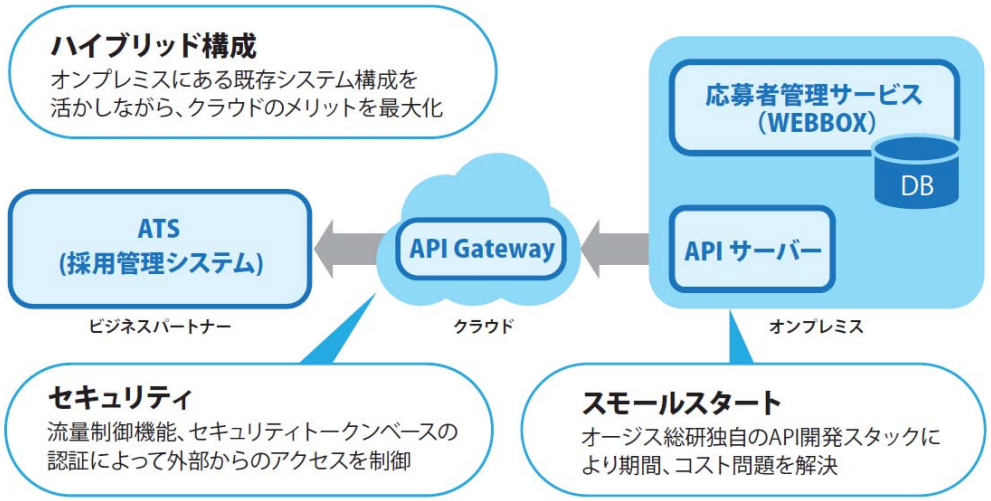


オージス総研

事例概要

事例

- 株式会社インテリジェンス様
- アルバイト求人情報サービス「an」の法人顧客向けサービス向上のために「API公開支援ソリューション」を採用



背景・課題

公開版につき削除：詳細はお問い合わせください。

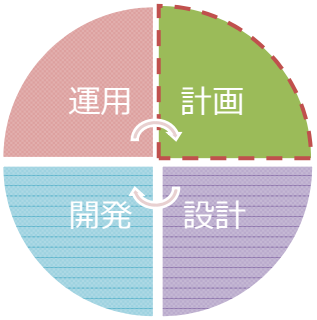
APIによる課題の解決

公開版につき削除：詳細はお問い合わせください。

オージス総研

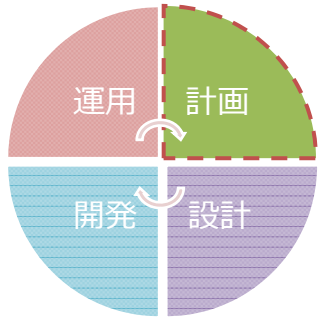
事例におけるAPI公開プロセス

事例における計画フェーズ



- API公開の目的を明確にする
  - 応募情報のデータを共有する
- APIの利用者および公開範囲を明確にする
  - **グループ内**のATS(採用管理システム)
- APIとして公開するデータ、サービスを明確にする
  - 求人クライアントが閲覧する**応募情報の取得**、**応募情報ステータスの更新**
- APIの利用シナリオを明確にする
  - 求人クライアントが応募者一覧を確認する → **応募情報参照系API**
  - 求人クライアントが応募ステータスを変更する → **応募情報更新系API**

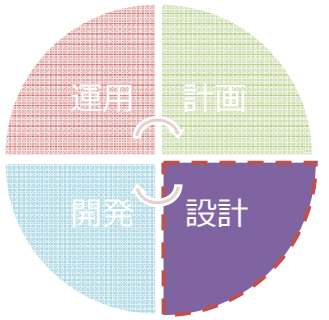
API以外のポイント

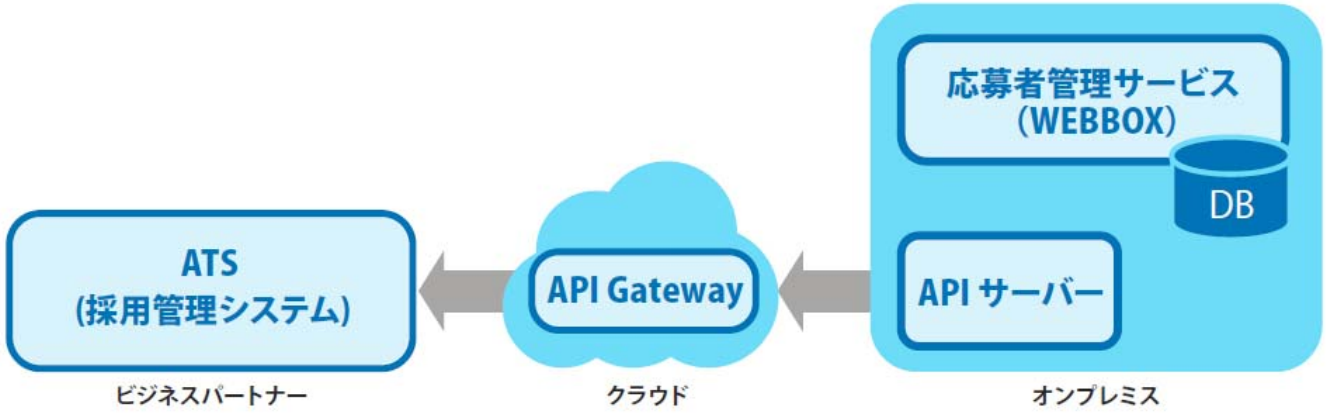


- スモールスタート
- 現行システムへの影響をできるだけ小さくする
- セキュリティ
- 将来に向けて拡張できるようにする

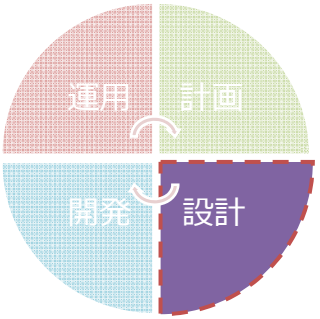
事例における設計フェーズ:ハイブリッド構成

- API Gateway
  - 認証
  - 流量制御
- APIサーバ
  - HTTPリクエスト/レスポンスのハンドリング
  - DBアクセス

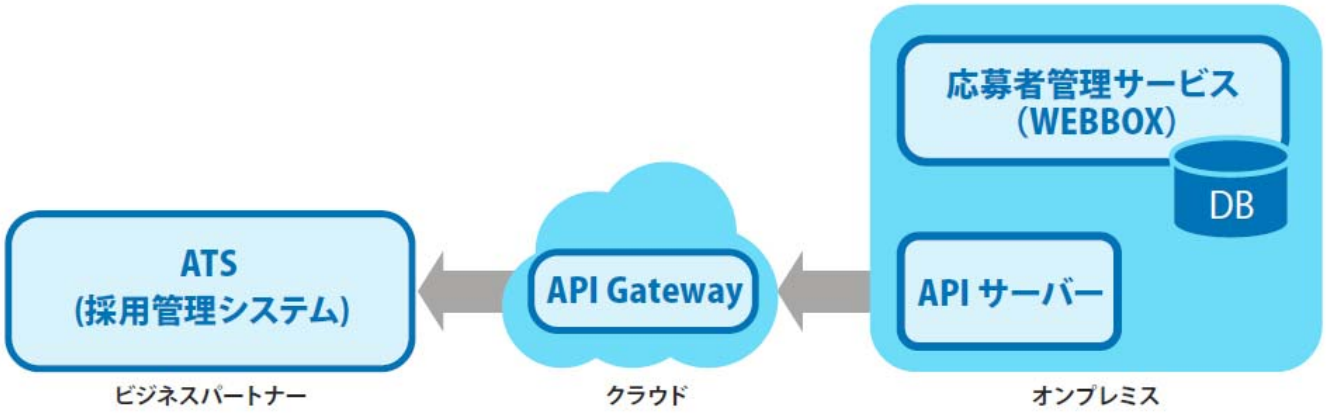




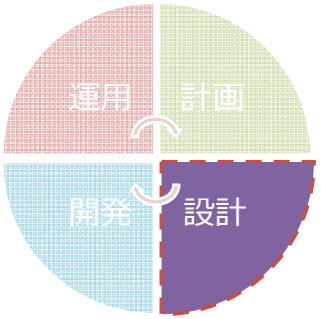
セキュリティ：ネットワーク



- パブリックサービスであるAPI Gatewayから保護領域にあるDBへいかに安全に接続するか。
  - 環境のセキュリティポリシー



セキュリティ: APIの認証・認可



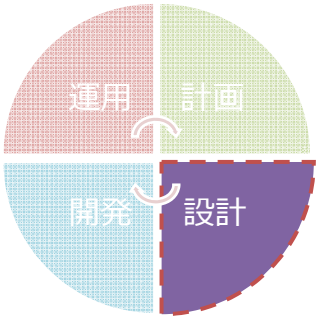
認証・認可

- “何に”に対して認証・認可するのか
  - → **クライアント(ビジネスパートナーの採用管理システム)** ← 事例ではこの考え方を採用
  - → クライアントのエンドユーザ(求人クライアント)
  - → クライアント + エンドユーザ

クライアントにクレデンシャル情報を保持できるかどうか

- 信頼できるクライアントなのかどうかが重要
  - → **信頼できる：グループ内のサーバ** ←事例ではこの考え方を採用
  - → 信頼できない： 広く配布されるモバイルアプリ、ブラウザのJSクライアント

セキュリティ：APIの認証・認可



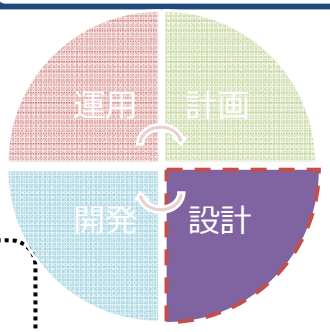
◘ 技術的選択肢

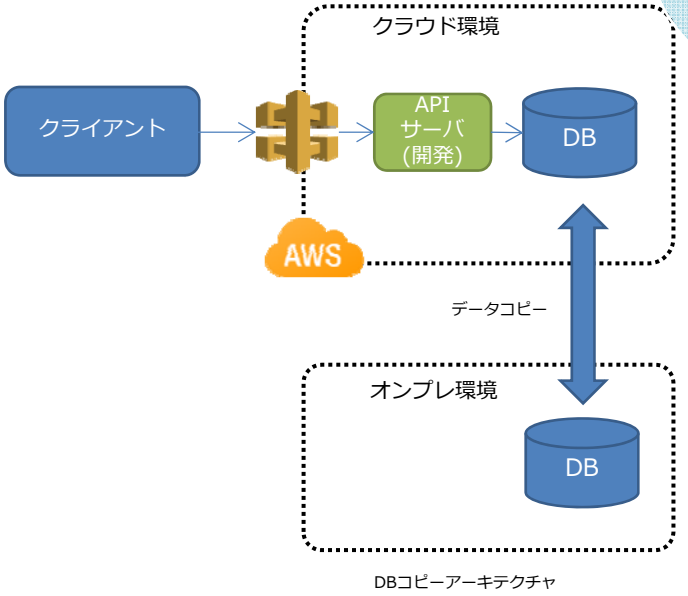
- OAuth2.0
- Basic 認証
- APIキー
- リクエストの電子署名
- SSLの相互認証
- IAM認証 Amazon Signature v4

→ 認証のアクセスキーを利用してリクエストに署名をする

現行システムへの負荷軽減/レイテンシー

- 現行システムへの影響をできるだけ小さくする
  - **流量制御(レートリミット)**
  - **キャッシュ**
  - **１レスポンスあたりのサイズ制限**
- レイテンシー
  - キャッシュ
    - → API Gatewayのキャッシュ
    - → **APIサーバのキャッシュ**←事例はこれを採用
    - → DBのキャッシュ
  - DBデータコピーアーキテクチャ

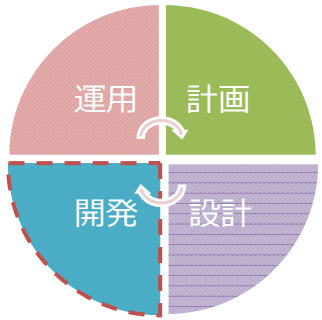


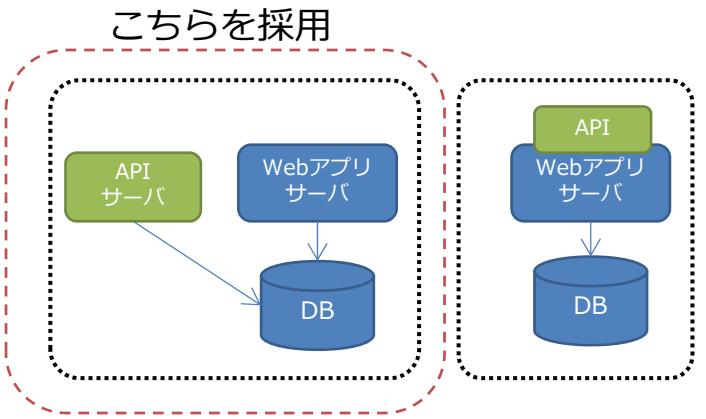


DBコピーアーキテクチャ

事例における開発フェーズ

- APIをどのように実装するか。
  - 既存WebアプリにAPIを追加
  - **既存DBを利用しAPIを新規開発** ←事例はこれを採用
- Java/Spring ベースのAPI開発スタック
  - 様々なOSSを組み合わせたAPI開発に必要な機能を持つフレームワーク
    - → **HTTP リクエスト/レスポンスのハンドリング**
    - → **メッセージバリデーション**
    - → **データベースアクセス**
    - → **キャッシュ**





応募情報参照系API

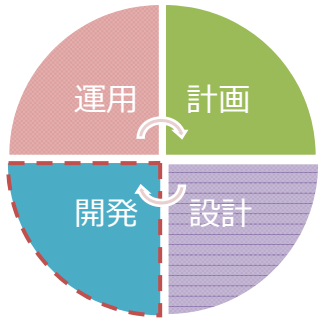
- 参照系APIのポイントは、フィルター、ソート、ページネーション
  - 必須のパラメーター、オプションのパラメーター
  - デフォルト値

- 応募情報参照系API
  - 入力
    → クライアントを特定する情報、期間
  - 出力
    → 応募情報
    →ページング情報

- 参照系APIの処理の流れ
  - HTTPリクエストのバリデーション
  - データセットを構成するDBへのクエリ
  - ユーザ視点のデータセット(JSON)の組み立て
  - HTTPレスポンスの返却

応募情報更新系API

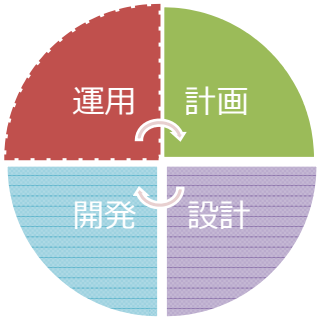
- 更新系APIのポイントはべき等性と副作用。同一リクエストの再送信を正しく処理できるようにする
  - べき等性: 同じリクエストは同じ結果になる
  - 副作用: リソースの変更が起こる



- 応募情報更新系API
  - 入力
    - → 応募情報を特定する情報
    - → ステータス
  - 出力
    - → 処理結果

- 更新系APIの処理の流れ
  - HTTPリクエストのバリデーション
  - データセットを構成するDBへ更新
  - ユーザ視点のデータセット(JSON)の組み立て
  - HTTPレスポンスの返却

事例における運用フェーズ:バージョン、リリース管理



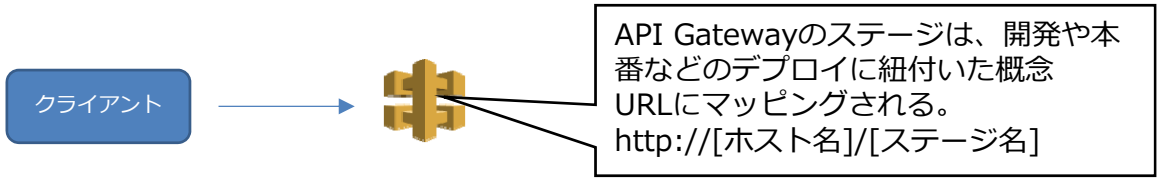
APIのバージョン、リリース管理

- 外部インタフェース – Amazon API Gateway の”ステージ”によって管理

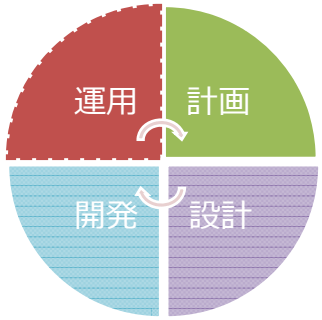
  →Swagger ファイルをバージョン管理システムで管理

- API 実装 – ソースコードをバージョン管理システムで管理

  →リリースは手順書をベースにした手動管理



APIのユーザ、契約管理



◘ APIのユーザ、契約管理

- **ユーザ、アクセスキー**

  → AWSの機能を使って実現

  → API 専用のロールを割り当て

- 契約管理

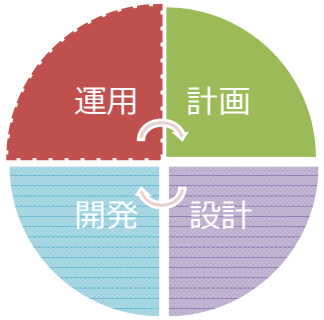
  → 事例では実施せず

APIのユーザ向けサポート

オージス総研

◘ APIのユーザ向けサポート

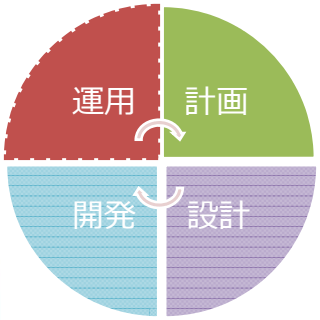
- **静的APIドキュメント(PDF)**
- 動的APIドキュメント(SwaggerUI)
- SDK



目次例

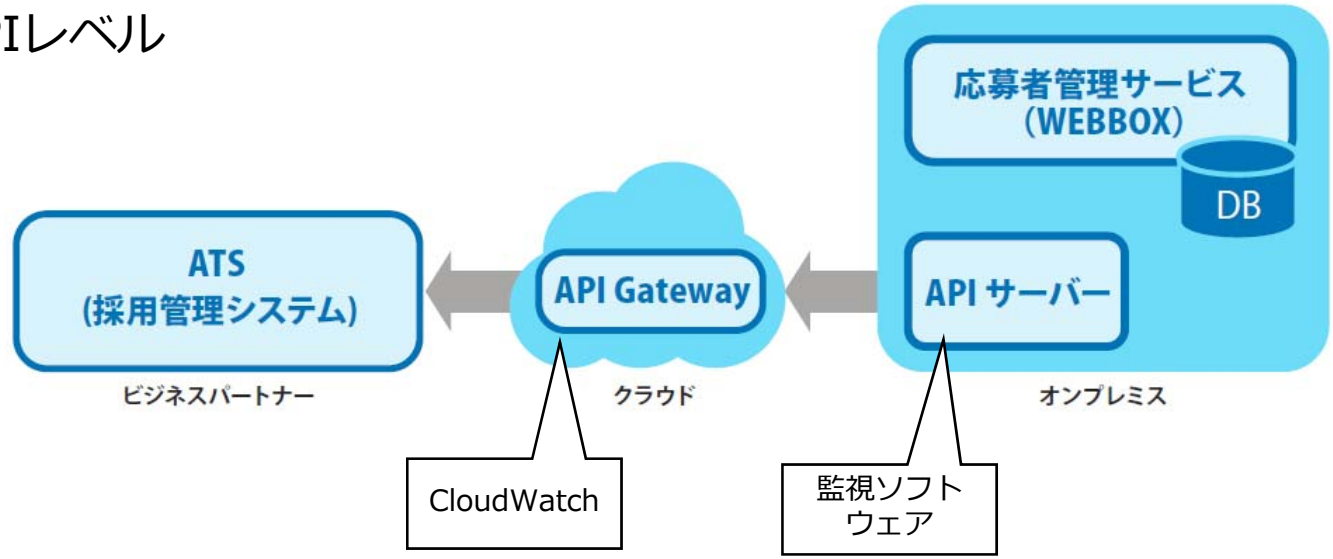
1. はじめに
2. API概要
3. ユースケース
4. API共通仕様
    1. スキーマ
    2. エンドポイント(Base URI)
    3. HTTPレスポンスコード
    4. 認証
5. 取得系API
    1. リクエスト
    2. レスポンス
6. 更新系API
    1. リクエスト
    2. レスポンス

APIの監視、障害対応



◘ APIの監視、障害対応

- サーバレベル
- ネットワークレベル
- APIレベル



まとめ

- API公開のプロセスをご紹介
  - APIの利用者を想定し、様々な角度から検討する
  - 基礎となるアーキテクチャを構築する
  - 一度きりのプロセスではなく、継続的なサイクルを実施する
- スモールスタートのAPI公開事例をご紹介
  - クラウドサービスを活用したハイブリッド構成
  - 既存システムへの影響を小さくするための方策
  - 参照系、更新系APIのポイント

オージス総研

当社APIソリューションご紹介

API公開支援ソリューション

オージス総研のクラウドとAPI開発・運用の知見を集めたサービスで、
「短期間でAPI構築」・「信頼性の高いAPIインフラ」を提供します

こんなお客さまに最適です

やりたいこと

- 新たなデータ販売チャネルを開拓したい
- ビジネスパートナーからのAPI対応の要望に応えたい
- モバイルやIoTデバイスを活用し、新しいビジネスモデルを構築したい

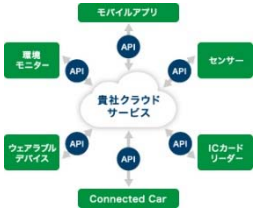
悩み

- 「APIの設計、実装」「運用・監視の設計」など技術面でのサポートが欲しい
- 最初の取り組みとして最小構成でスタートしたい

活用シーン

API提供により、ビジネスパートナーのサービスとデータを連携する

API活用により、様々なデバイスやアプリとサービスを連携する



サービスメニュー

- API構築サービス

APIプラットフォーム構築	クラウド、ハイブリッド、オンプレミスに対応 要件にマッチした製品/サービスを利用してご提供。 AWS、IBM API Connect、Anypoint Platform
API開発	API開発のベストプラクティスを詰め込んだAPI開発スタックを使い、効率的にAPIを開発

- API運用サービス

APIアップデート	APIのデータ項目の追加などAPIのアップデート、リリースの実施
APIプラットフォームメンテナンス	定期的なAPIプラットフォームのセキュリティアップデート
APIプラットフォーム監視	APIプラットフォームの障害・異常検知および通知
API障害分析、対応	障害発生時の原因調査、切り分けおよび復旧対応
API利用状況レポート	APIの利用状況のレポーティング
APIユーザサポート	API利用者への問い合わせ対応、APIクライアント開発支援

Amazon API Gateway とは

ロ　どのようなスケールであっても、簡単に API の作成、配布、保守、監視、保護が行える、AWSサービス

できること

- **低コストで効率的な利用**
  - APIに対する呼び出しと、送出されるデータに対してのみ料金が発生
- **どのようなスケールにも対応**
  - エッジロケーションの世界的ネットワークを活用し、可能な限り低いレイテンシーでエンドユーザーに提供できる。
  - スロットリング、API呼び出しに対する出力のキャッシュなどによりバックエンドの保護ができる

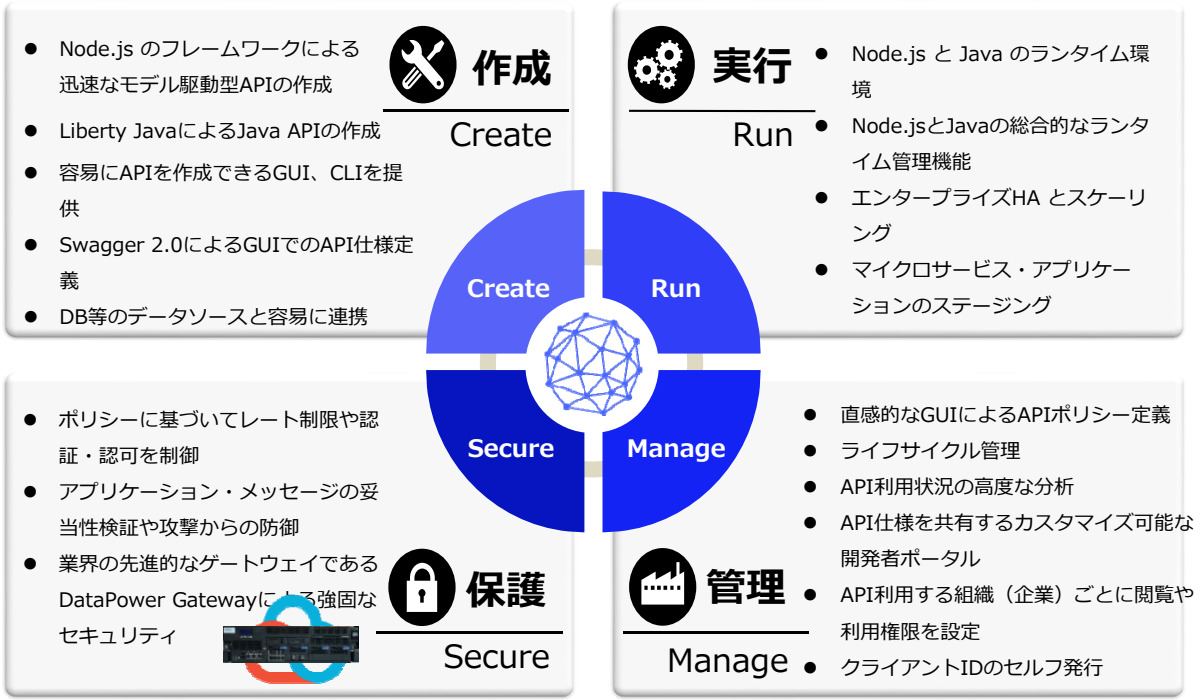
他のAWSサービスとインテグレーションすることで可能になること

- **APIアクティビティの容易な監視**
  - Amazon CloudWatchによってサービスの呼び出しを視覚的にモニタリングできる
    （パフォーマンスのメトリックスとAPI 呼び出しについての情報、データのレイテンシー、エラー率を確認できる）
- **柔軟なセキュリティ管理**
  - AWS Identity and Access Management (IAM)、Amazon Cognito といったAWSの管理ツールやセキュリティツール
    を使用して、APIに対するアクセス認証が実施できる

https://aws.amazon.com/jp/api-gateway/から抜粋

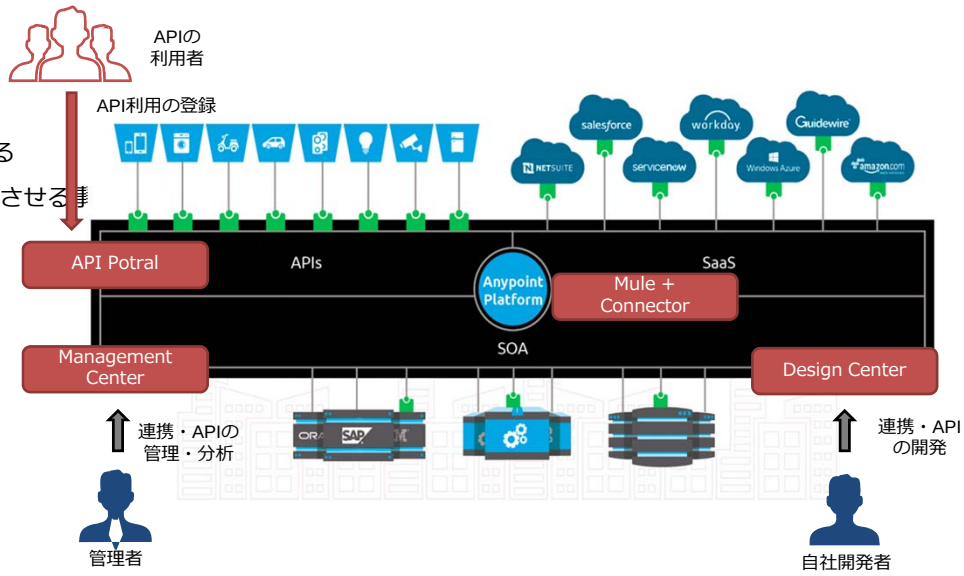
IBM API Connect とは

- APIの作成、実行、管理、保護の機能を包括的に提供し、API公開を実現する



Anypoint Platform™ とは

- API利用、API公開どちらにも利用可能な連携プラットフォームを提供する
- 一つの基盤で、クラウド・オンプレミスを問わず、全てのアプリケーション・データソース・APIとの接続を可能にするエンタープライズ向け連携プラットフォーム
- 特長
  - APIの設計、連携アプリケーションの実装および、統合的な管理ができる
  - 一度開発した連携アプリは、オンプレとクラウドどちらの環境でも稼働させる事
  - APIドキュメントやAPIのユーザー登録を行うAPIポータルの公開も可能



お問い合わせ先

株式会社オージス総研

営業本部営業企画部

【 TEL 】03-6712-1201（東京） / 052-209-9390（名古屋） / 06-6871-8054（大阪）

【 E-mail 】info@ogis-ri.co.jp

【 URL 】http://www.ogis-ri.co.jp