プラットフォームエンジニアリング

SAチーム向けAOAIのAPI公開と活用例

Agenda

- 自己紹介
- プラットフォームエンジニアリングの概要
- SAチーム向けAOAIのAPI公開
  - API ManagementとAOAI
    - API構成
    - API実装の工夫点
  - 活用事例

自己紹介

■ ハンドルネーム：風の時代のエンジニニャ
■ 主な業務
- プラットフォームエンジニアリング
  - プラットフォーム設計構築
  - セキュリティ施策
- SRE
  - 監視改善(ダッシュボード作成など)
  - IaCリファクタリング

■ 趣味
タロット(ほぼ毎日占っています)

■ SNS/Githubアカウント
- X：@windagecat
- Github：https://github.com/windagecat

テーマのきっかけ(イントロダクション)

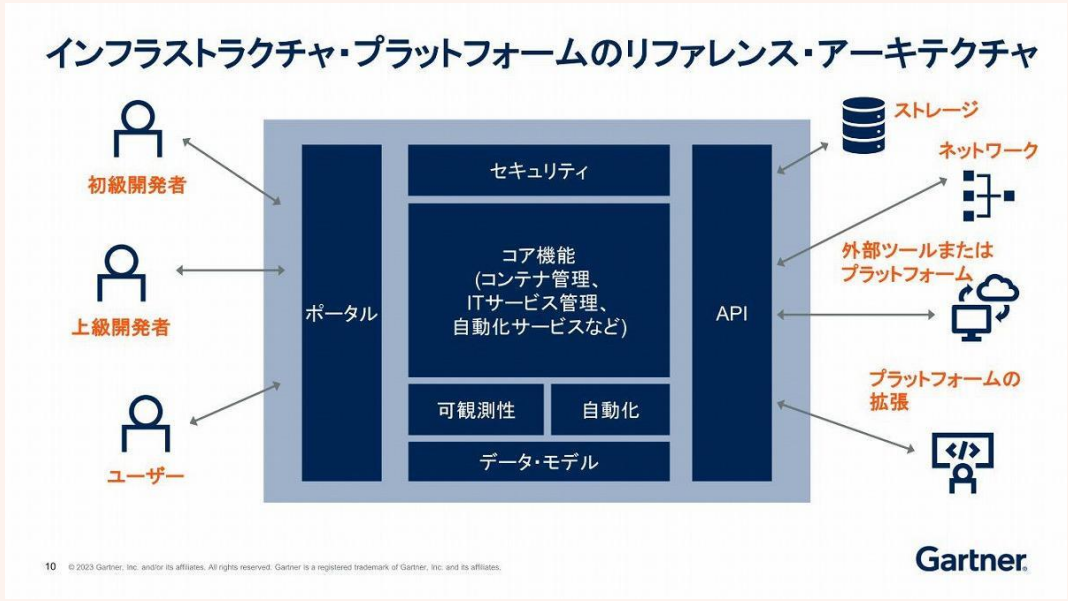
プラットフォームエンジニアリングの業務に携わる中で、開発チーム(SAチーム)向けにどういうサービスを提供したらいいかと考えた時に、これからはDevOpsにAIがどんどん活用されていくと思い、個人的な趣味で、Azure Open AI Service(AOAI)のAPI公開について検証してみました。
今回は、検証の中で気づいたポイントや工夫点などを中心にご紹介します。


プラットフォーム
エンジニアリングの概要

プラットフォームエンジニアリングの概要

教科書通りの説明

開発者にセルフサービス型で、プラットフォームの構築と運用を提供するサービスおよび専門分野を指す。
あらかじめ決められたテンプレートを提供することで、開発者の認知負荷を軽減し、開発コーディングになるべく集中できるようサポートする。



参考：[プラットフォーム・エンジニアリングとは何か？ガートナーが解説するDevOps進化の要点]

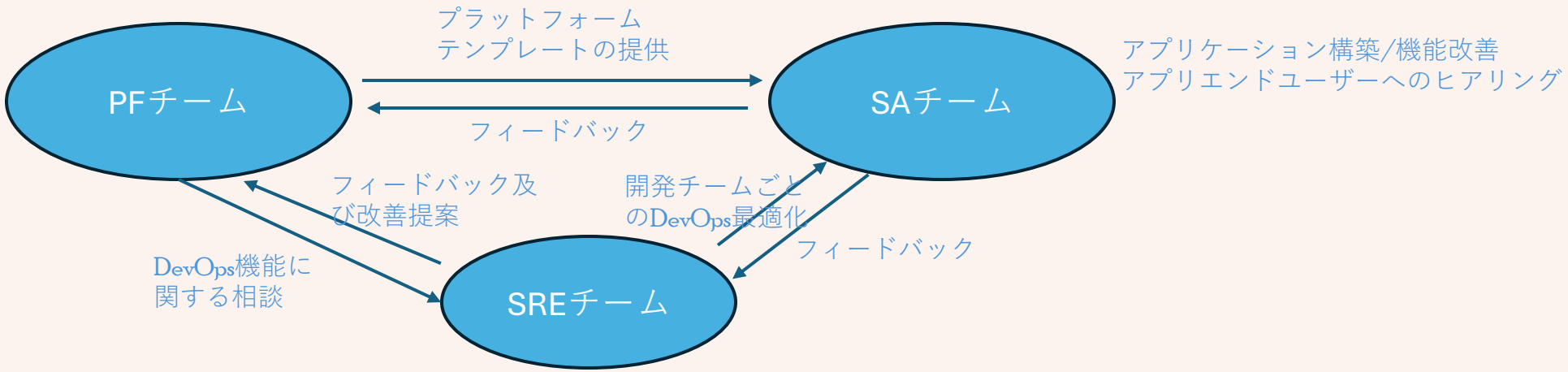
プラットフォームエンジニアリングの概要

もう少し踏み込むと・・・

プラットフォームエンジニアリングでは、以下のチームに分かれる。

- **プラットフォームチーム(PFチーム)**：開発チーム向けにプラットフォームテンプレートを提供
- **開発チーム(SAチーム)**：アプリケーション構築/機能改善
- **SREチーム**：DevOps最適化

これらのエンジニアチームが、フラットな関係で連携していくための仕組みのことでもある。



プラットフォームエンジニアリングメリット

(Copilotの回答)

- **内部開発者プラットフォームの構築**: 組織内で標準化された開発環境を提供することで、開発者が効率的に仕事を進めることができます。
- **デリバリープロセスの標準化とセキュリティ確保**: 重要なデリバリープロセスを標準化することで、セキュリティリスクを最小限に抑えることができます。
- **サービスレベルアグリーメントの設定と順守**: 社内のサービスレベルを明確にし、それを順守することで信頼性が向上します。
- **チームのパフォーマンスメトリクスの監視**: チームの成果を監視し、パフォーマンスを向上させるためのデータを提供します。
- **開発者エクスペリエンスと生産性の向上**: 開発環境が整備されていることで、開発者は自分の仕事に集中でき、全体の生産性が向上します。
- **再利用可能なコンポーネントとサービスの活用**: アプリケーションの構成要素を再利用可能な形で設計し、新しいプロジェクトでも効率的に活用できます。

要約すると、

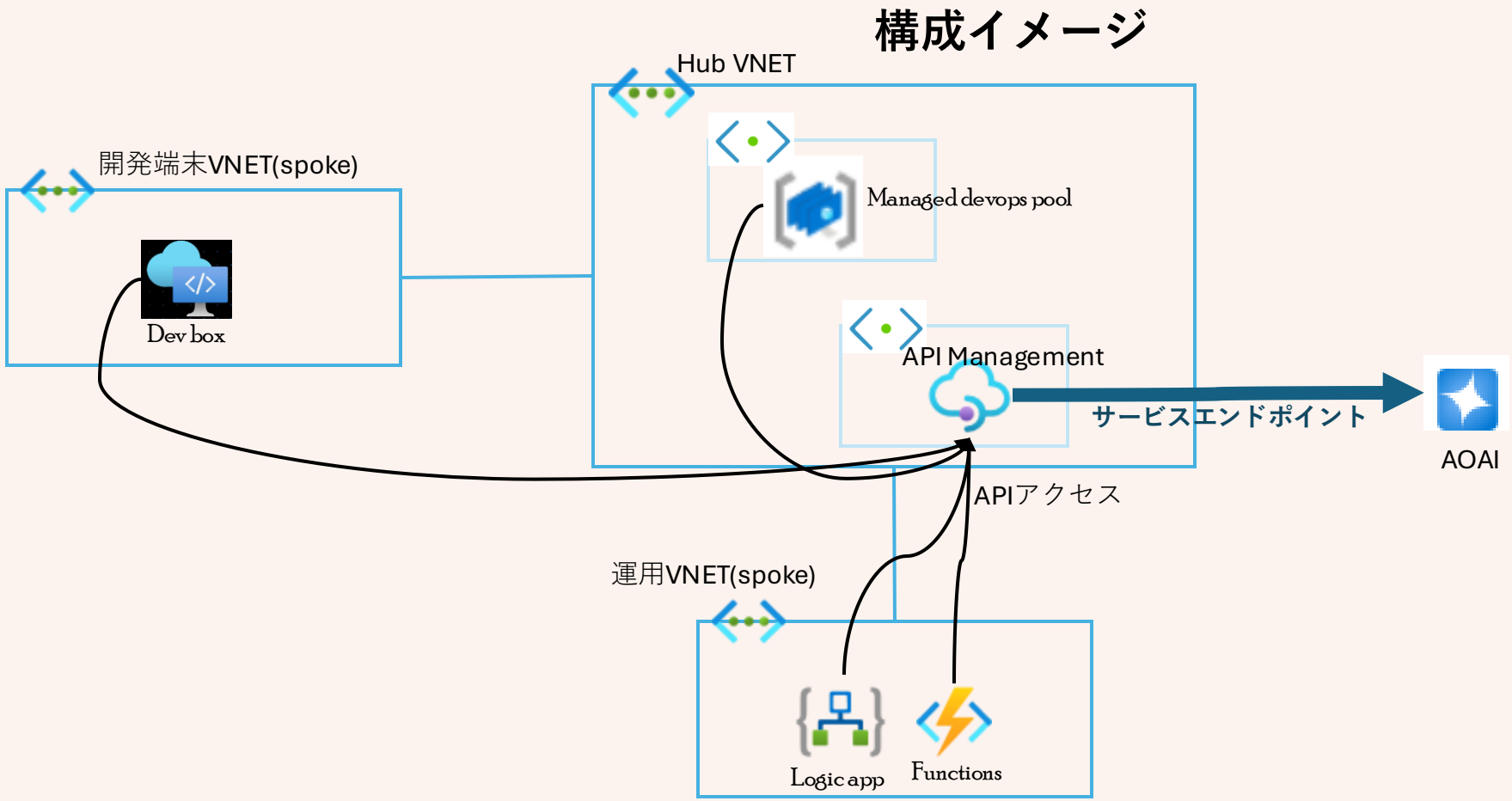
- 標準化による品質確保とセキュリティ強化
- 認知負荷軽減による開発者の生産性向上


SAチーム向け
AOAIのAPI公開

API Management と AOAI

API構成

AOAIをSAチーム向けに公開する場合、監視面や集約のメリットからAPI Managementを介したAPI公開を実装します。

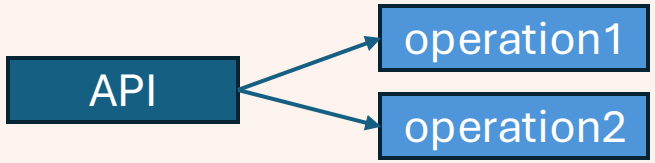


API実装の工夫点

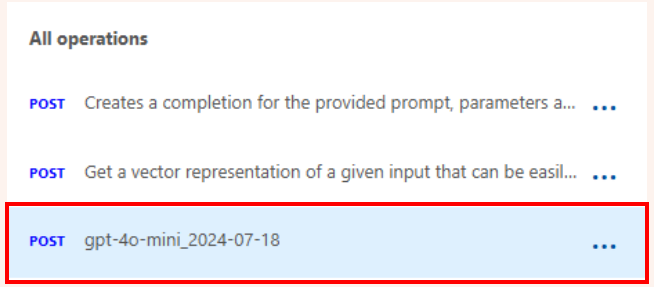
① AOAIのモデルバージョンごとに、APIのオペレーションを分ける
② AOAIのモデルバージョンの自動アップデートを無効にする
③ AOAIの負荷分散の構成
④ トークン制限の構成
⑤ 余計なAPIをブロック
⑥ 監視用ダッシュボードの実装
⑦ サブスクリプションの払い出し

API実装の工夫点

①AOAIのモデルバージョンごとに、APIのオペレーションを分ける



例）GPT-4o-miniの20240718のバージョン(gpt-4o-mini_20240718)
GPT-4oの2024-08-06のバージョン(gpt-4o_20240806)
GPT-4oの2024-11-20のバージョン(gpt-4o_2024-11-20)

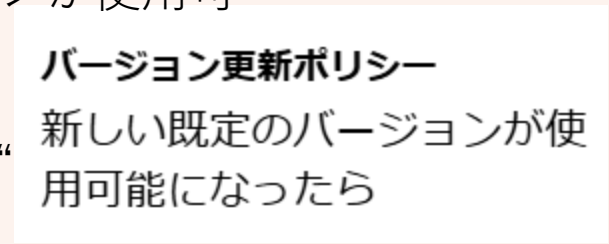


モデルバージョンごとにオペレーションを分けることで、
開発者のカスタマイズ性が上がる

API実装の工夫点

②AOAIのモデルバージョンの自動アップデートを無効にする

AOAIのモデルをデプロイすると、デフォルトでは新バージョンが使用可能になったら自動アップデートされる。
・デフォルト更新ポリシー
Properties.versionUpgradeOption:"OnceCurrentVersionExpired“



バーションが上がると挙動が変わり開発者に影響する可能性があるので、更新ポリシーを「NoAutoUpgrade」にして、自動アップデートを無効化するとよい。
前のページで説明したように、バーションごとにAPIのオペレーションを分けて管理するとよい。

- Terraformでの自動アップデートを無効化方法

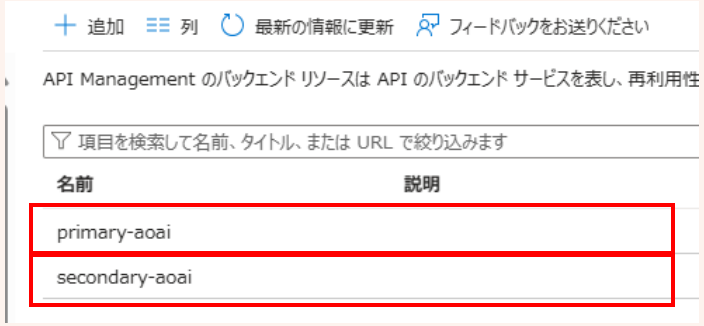
例）

```
resource "azurerm_cognitive_deployment" "example" {
  name                 = "example-cd"
  cognitive_account_id = azurerm_cognitive_account.example.id
 version_upgrade_option = “NoAutoUpgrade”
  ....
```

API実装の工夫点

③AOAIの負荷分散の構成

AOAIをAPIMのバックエンドに登録して、apiポリシーでランダム数値で負荷分散するようにしている




```
<fragment>
    <set-variable name="urlId" value="@(new Random(context.RequestId.GetHashCode()).Next(1, 101))" />
    <choose>
        <when condition="@(context.Variables.GetValueOrDefault<int>("urlId") < 51)">
            <set-backend-service backend-id="primary-aoai" />
        </when>
        <when condition="@(context.Variables.GetValueOrDefault<int>("urlId") > 50)">
            <set-backend-service backend-id="secondary-aoai" />
        </when>
        <otherwise>
            <return-response>
                <set-status code="500" reason="InternalServerError" />
                <set-header name="Microsoft-Azure-Api-Management-Correlation-Id" exists-
action="override">
                    <value>@{return Guid.NewGuid().ToString();}</value>
                </set-header>
                <set-body>A gateway-related error occurred while processing the request.</set-body>
            </return-response>
        </otherwise>
    </choose>
</fragment>
```


API実装の工夫点

④トークン制限の構成

azure-openai-token-limit　使ってトークン制限を構成(従来では、rate-limitやリクエストボディサイズを制限していた)

サブスクリプションやIPアドレスごとにトークン制限が可能。

また、このポリシーの中にTPM(tokens-per-minute)の設定があるが、どれぐらいトークン数に制限すればいいか迷う場合は、一旦はAOAIのTPM数に合わせて設定しておいて、ダッシュボードで数か月間TPMを観察し、調整するとよい。

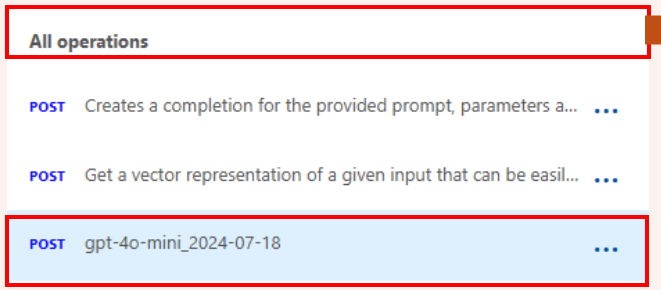
```
<azure-openai-token-limit tokens-per-minute="30000" counter-key="@(context.Subscription.Id)" estimate-prompt-tokens="true" tokens-consumed-header-name="consumed-tokens" remaining-tokens-header-name="remaining-tokens" />
```


API実装の工夫点

⑤余計なAPIをブロック

使用しないAPIは、400エラーを返すようにAll operationsスコープでポリシーを設定
使用するAPIは各オペレーションごとにポリシーを設定

使用しないAPIはAll operationsスコープで400エラーを返すように設定




```
<inbound>
      <!-- 既定ではエラーで折り返す -->
      <return-response>
          <set-status code="400" reason="Bad Request" />
          <set-body>{
          "error": {
              "code": "OperationNotSupported",
              "message": "Your request operation is not
supported"
          }
       }</set-body>
      </return-response>
  </inbound>
```

使用するapiはオペレーションごとのポリシーで設定

```
<policies>
  <!-- Throttle, authorize, validate, cache, or transform the requests -->
  <inbound>
    <authentication-managed-identity resource="https://cognitiveservices.azure.com" />
    <include-fragment fragment-id="backend-aoai" />
    <azure-openai-emit-token-metric namespace="AzureOpenAI">
      <dimension name="API ID" />
      <dimension name="Subscription ID" />
    </azure-openai-emit-token-metric>
    <azure-openai-token-limit tokens-per-minute="30000" counter-key="@(context.Subscription.Id)" estimate-prompt-
tokens="true" tokens-consumed-header-name="consumed-tokens" remaining-tokens-header-name="remaining-tokens" />
...
```

API実装の工夫点

⑥監視用ダッシュボードを作成

監視用に便利なダッシュボードを作成してみました！
(抜粋)

- APIM 平均 Capacity
- AOAIごとの1時間あたりの平均応答時間（秒）
- 1時間あたりのOpenAI APIのオペレーションごとのResponse 429の回数
- サブスクリプションごとの1分間あたりの合計プロンプトトークン/完了トークン/合計トークン数
- サブスクリプションごとの1時間あたりのリクエスト数

API実装に関する工夫点

⑦サブスクリプションの払いだし

サブスクリプションを払い出す場合、SAチームごとに払い出すか/ユーザーごとに払い出すかのどちらかになる。
どちらを採用するか状況によって決めるとよい。

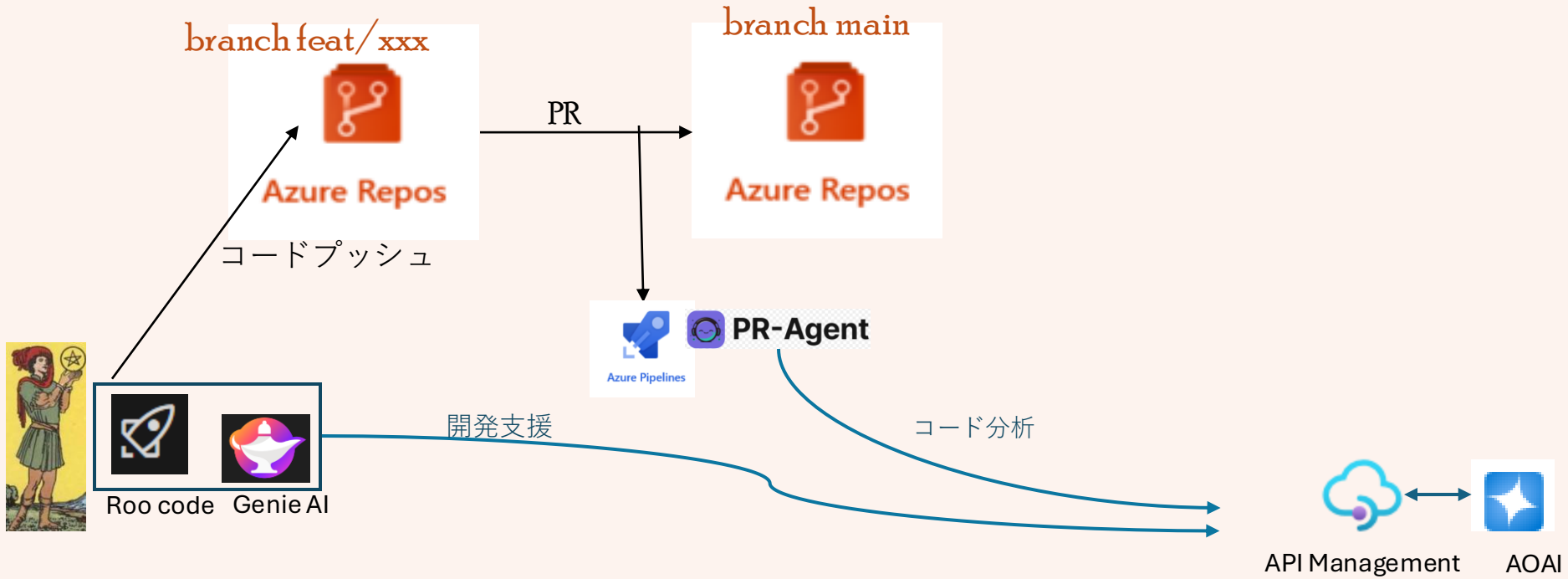
監視の注意点として、
SAチームごとに払い出すケースで、ユーザーごとのリクエスト数やトークン数を監視したい場合はIPアドレスごとに集計するとよい。
トークン数について、azure-openai-emit-token-metric **ポリシーでカスタムメトリック**(IPアドレスなど)ごとのトークン数をapplication insightsに送信可能。
しかし、この機能は現在プレビュー(2025/2現在)で、カスタムメトリック数により出力されるログ数に上限がある。

APIの活用例

APIの活用例①コードレビュー

AOAIのAPIを使ってコードレビューができる。
コードレビューに使えるツールについて、紹介します。

- PR-agant：CICDでコードレビューが可能。レビュー結果をPRコメントとして出力できる
- Roo-code(Roo-cline)やGenie AI： Clineをフォークして、機能追加されたAI agentツール



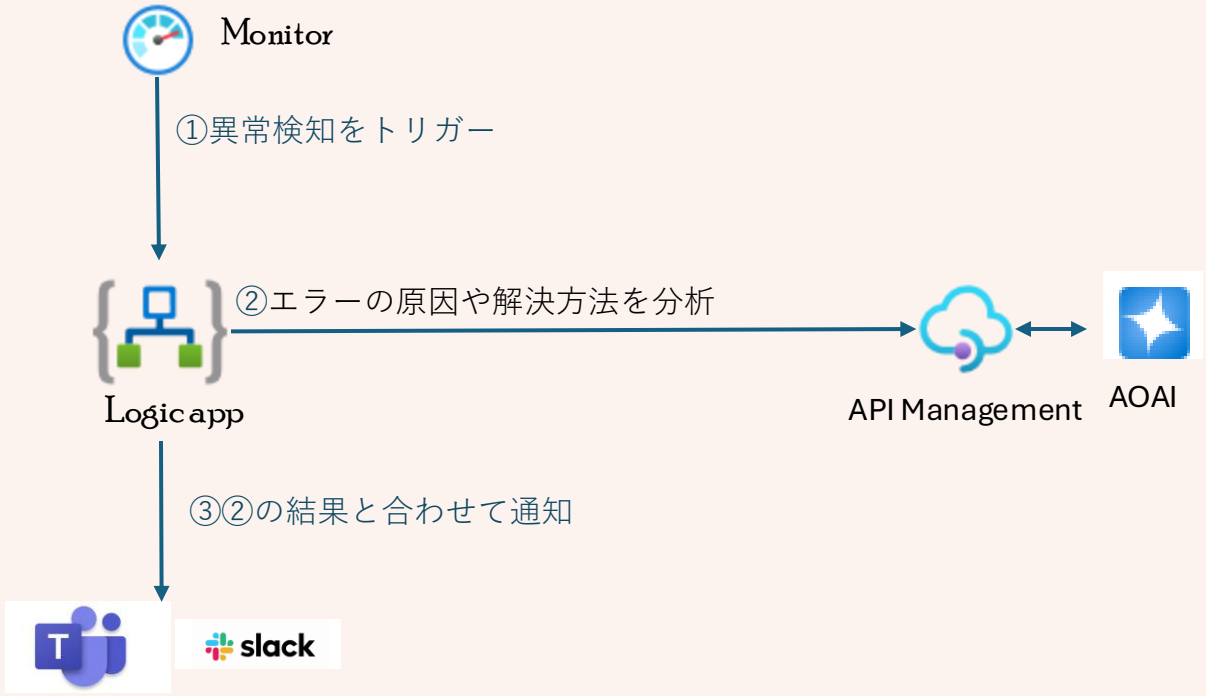
APIの活用例②NoOps

NoOpsとは

できるだけ運用管理を自動化することで、運用の手間を解消するエンジニアリング手法。

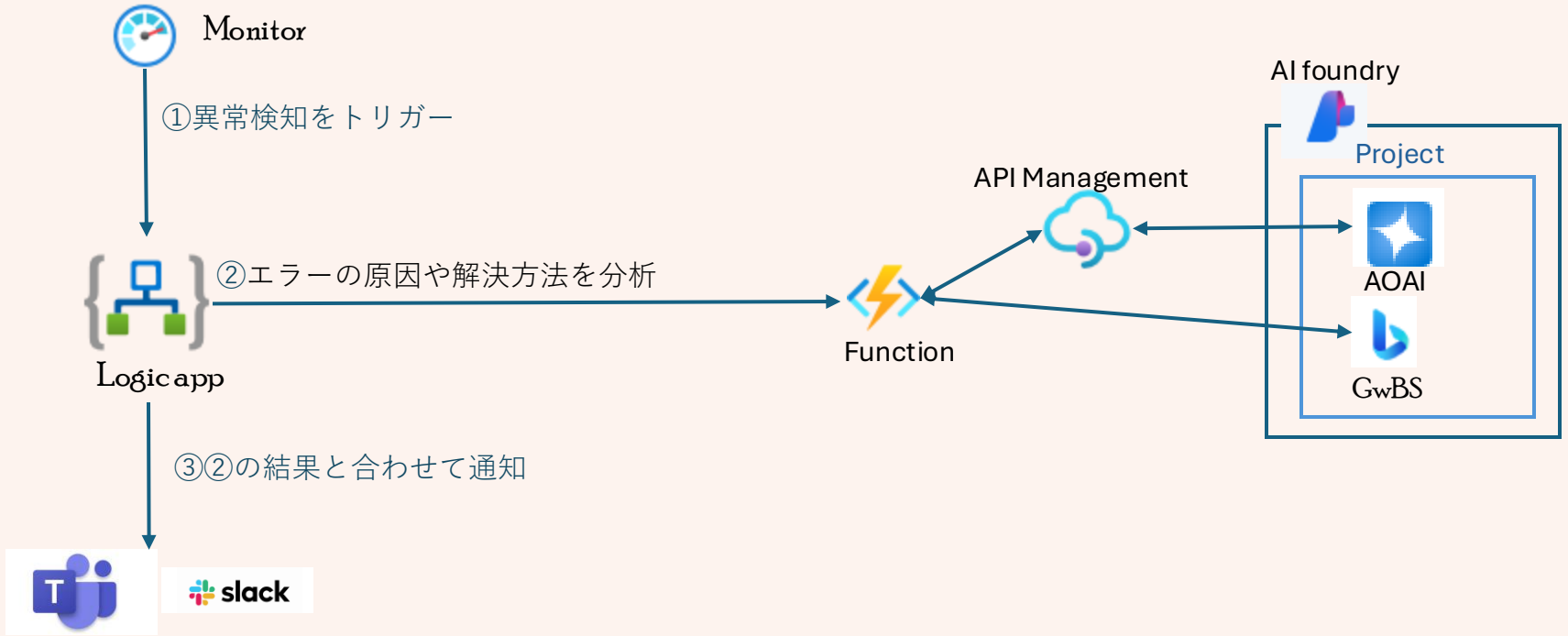
活用例

- 異常検知で、エラーの原因や解決方法の提示



APIの活用例②NoOps　補足

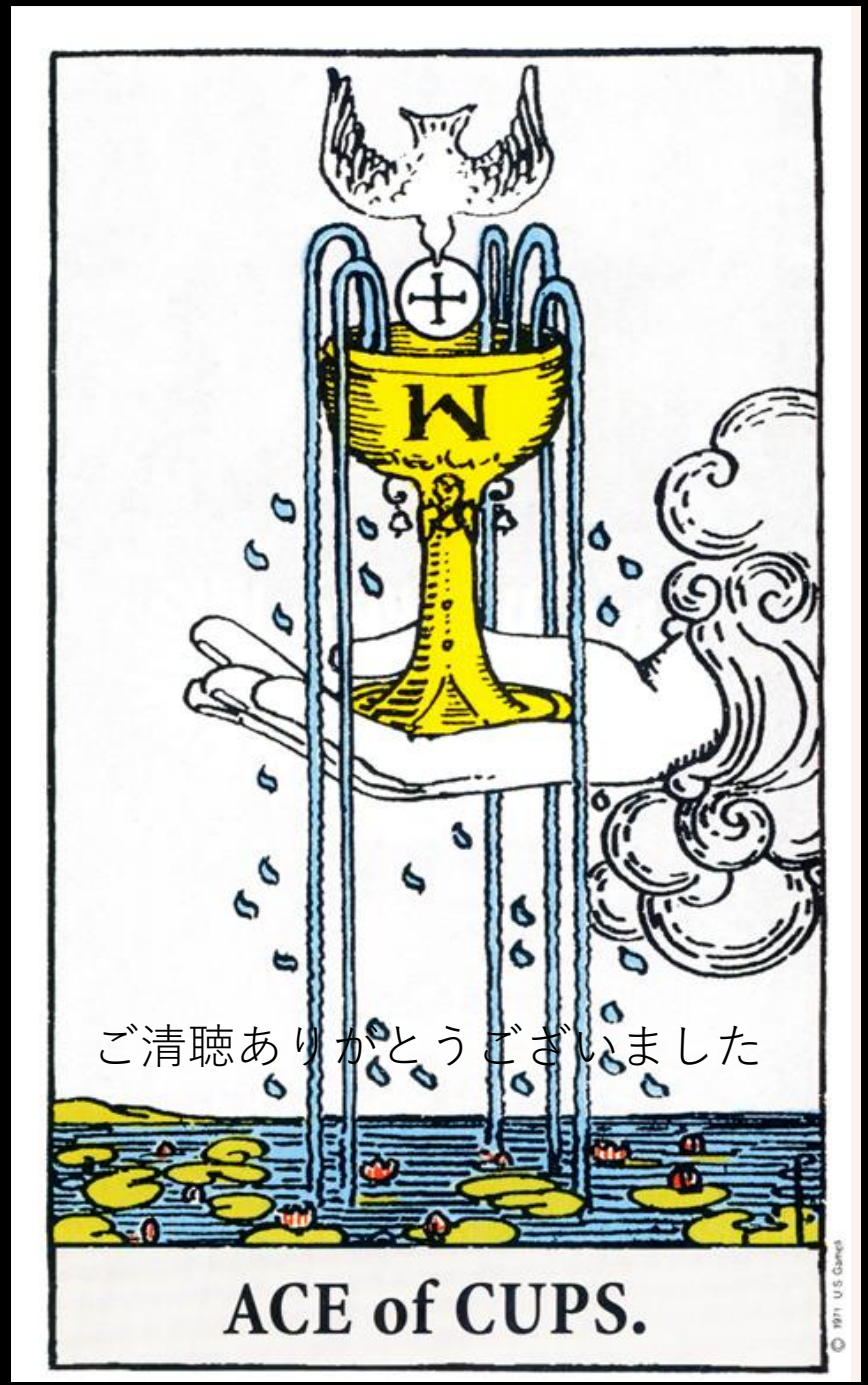
AOAIは常に最新の学習データを保持しているわけではないので、最新情報に基づいて回答結果を得るには、Grounding with Bing Search(GwBS)※1を活用するとよい。



※1 Bing Search は、2025/3/6に廃止

最後に

テスト用の API 構成のデプロイについて、
Githubで公開(https://github.com/windagecat/SA-AOAI-APIM)しています。
ぜひ試しにデプロイしてみてください！



ご清聴ありがとうございました