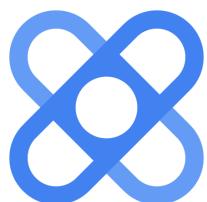


# Apigee API Jam

## API 管理の基礎 - 2024

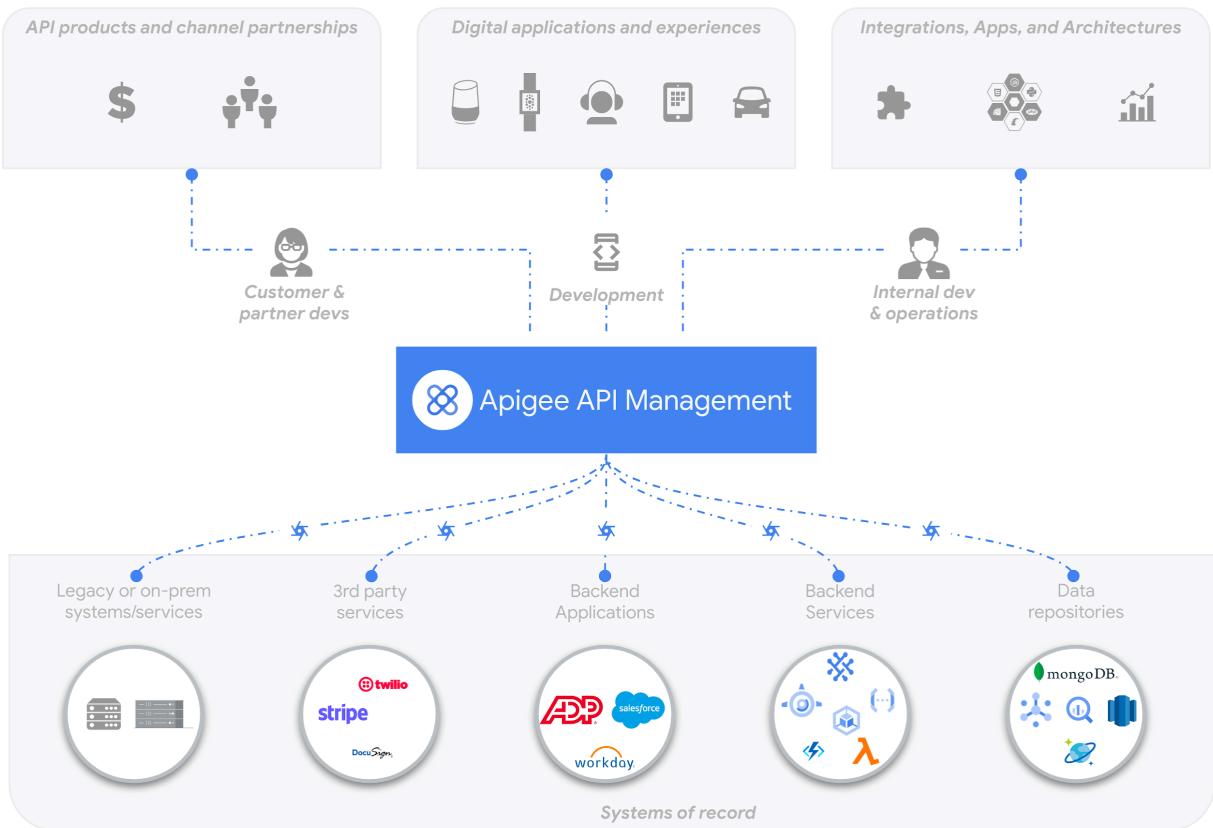
日本語翻訳資料

Google Cloud Japan  
Last Update : Mar 1, 2024



# はじめに

Apigee API 管理の基礎を学ぶ Google の Apigee API Jam へようこそ。このハンズオンラボでは、基本的な API 管理の概念と Apigee API 管理プラットフォームについて紹介します。適切に管理された開発者エコシステムで成功する API プログラムを構築する方法を俯瞰的に理解することに重点を置いています。このラボでは、API 設計、API セキュリティ、開発者の利用、API 分析とモニタリングなど、API ライフサイクルのすべてのフェーズにわたって API を管理する方法を学びます。



ここにあるすべてのマテリアルは、[Apache 2.0 license](#)に基づいてリリースされています。

## ラボの目的

このラボでは、次のタスクを実行する方法を学習します:

1. API の設計と作成 - OpenAPI 仕様に基づいて API プロキシを設計および作成します。
2. API 製品、アプリ、API キー - API キーを使用して API を保護し、API を API 製品にバンドルし、アプリ、API 製品、API キー間の関連性を理解します。
3. レート制限と動的応答 - API 製品層の割り当てに基づいて API 消費を制御します。
4. アプリ開発者体験 - セルフサービス開発者ポータルで API 製品カタログを公開します。アプリ開発者のオンボーディング エクスペリエンスとセルフサービス API の使用をテストし、Teams と Audience の資格を使用してリソースへのアクセスを制限します。
5. API Analytics - Apigee Analytics を使用して API プログラムの成功を測定します。

## ラボの前提条件 - ツール

このラボでは次のものが必要となります:

- Apigee Platform UI にアクセスするための[最新の Chrome](#)などのモダンなウェブブラウザ
- [OpenAPI Specification](#) (旧Swagger) の基本的な理解
- [VS Code](#) など、任意のテキスト/コード エディター
- API をテストするための HTTP クライアント(cURL、[Postman](#) など)へのアクセス。アクセス権がない場合は、[Apigee Debug tool](#) を使用できます。

## ラボの前提条件 - アクセス

このラボでは、次の URL/ドメインにアクセスする必要があります。企業プロキシ/ファイアウォールに接続している場合は、ラボ期間中これらを一時的に (サブドメインを含む) 許可してください。

必須:

- [qwiklabs.com](https://qwiklabs.com)
- [googleapis.com](https://googleapis.com)
- [google.com](https://google.com)
- [apiservices.dev](https://apiservices.dev)
- [apigee.io](https://apigee.io)

オプショナル (ツールのダウンロードや参照リンクなど):

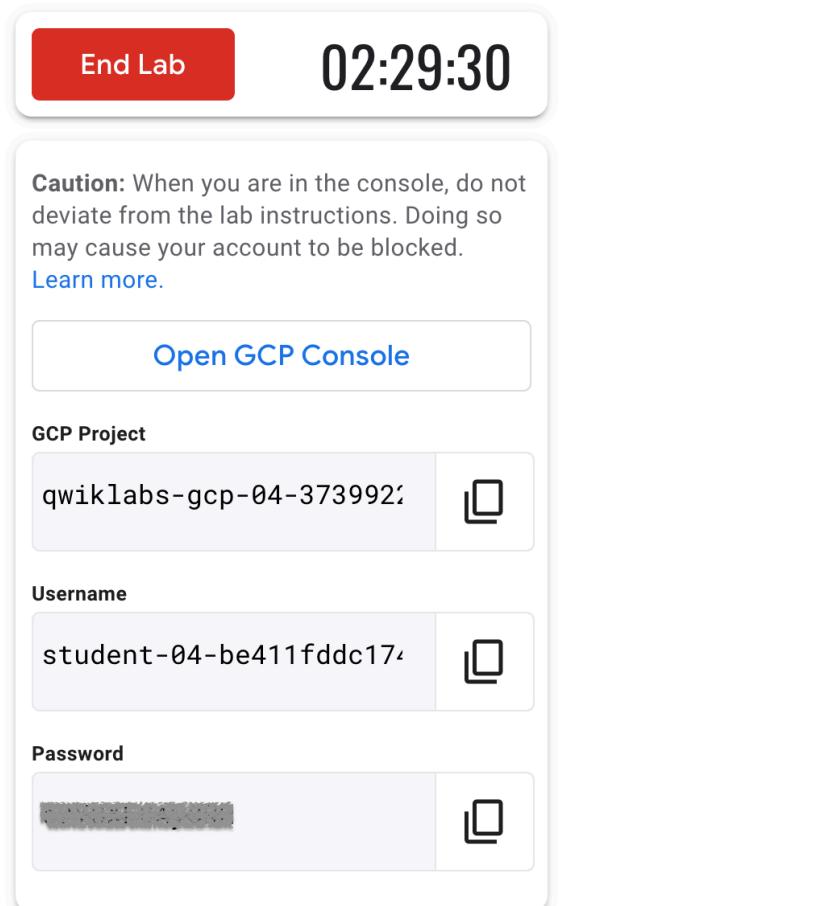
- <https://code.visualstudio.com/download>
- <https://www.postman.com/downloads/>
- <https://github.com>
- <https://openapi.tools/>
- [https://en.wikipedia.org/wiki/Same-origin\\_policy](https://en.wikipedia.org/wiki/Same-origin_policy)
- <https://youtu.be>
- <https://www.youtube.com>
- <https://www.apache.org>

## ラボのセットアップ

このラボでは、[Apigee 組織 \(Org\)](#) と、その組織が関連付けられている基盤となる Google Cloud Platform (GCP) プロジェクトにアクセスする必要があります。これを取得するには、このラボの左側のパネルにある [Start Lab] ボタンをクリックします。

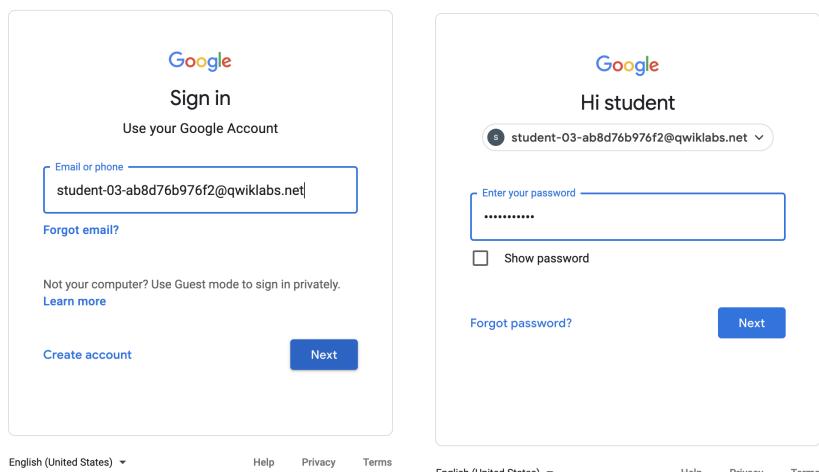


これにより、GCP 上にラボ用プロジェクトが生成され、Apigee 上に関連する評価組織が生成されます。この組織はこのラボの期間中利用できます。

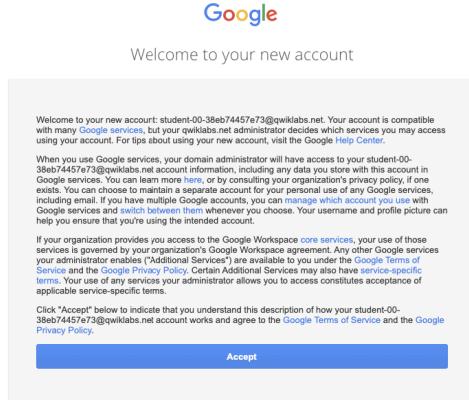


[Open GCP Console] を右クリックし、[シークレット ウィンドウでリンクを開く] を選択します。

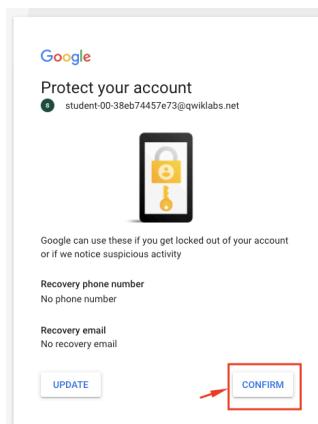
そこで、左側にあるラボの認証情報を使用してログインします。



次の画面で [Accept] をクリックして、Google 利用規約と Google プライバシー ポリシーに同意する必要があります。



次の画面では、アカウント保護設定の確認を求められる場合があります。



[Confirm] をクリックします。

その後、Google Cloud Console が表示されます。それ以外の場合は、新しいタブを開いて <https://console.cloud.google.com/> と入力し、Qwiklabs 環境で表示されている GCP プロジェクトと同じものが選択されていることを確認してください。

The screenshot shows two browser windows side-by-side. The left window is the Google Cloud Platform Dashboard for project 'qwiklabs-gcp-00-c674d76fa36c'. It displays various metrics and resources under sections like Project info, Compute Engine, Google Cloud Platform status, Resources, APIs, and Billing. The right window is the 'Apigee API Jam Lab 1: API Management Fundamentals' page, showing a login form with fields for GCP Project (set to 'qwiklabs-gcp-00-c674d76fa36c'), Username ('student-00-4644c819cc1e'), and Password. A red box highlights the 'GCP Project' dropdown in the Apigee UI.

これで、このラボを開始する準備が整いました!

## ラボのセットアップ (共有アクセス組織/環境のみ)

このラボによってプロビジョニングされた組織および Google プロジェクト ([Start Lab] をクリックすると、左側のパネルに表示されます) を使用している場合は、これを無視することを選択できます。

あなた(または監督者)が、他のユーザーと共有している既存の [Apigee 組織\(Org\)](#) の使用を選択する場合があります。その場合は混乱を避けるために、すべてのアセット名の先頭にイニシャルを付けてください。たとえば、API プロキシ名 = {your-initials}\_{プロキシ名} などです。

### 備考: Apigee UI と GCP Console

- Apigee は、近い将来導入される予定のポータルを除く、ほとんどの機能を [Apigee UI](#) から [GCP Console](#) に移行しました。
- 完全な移行が完了するまでは、両方の UI が共存します。
- このラボは、ポータル(パート 4)と分析の一部(パート 5)を除き、ほとんど GCP Console で実行します。



# パート 1 - OpenAPI 仕様を使用した API プロキシの設計と作成

ペルソナ: API チーム

## ユースケース

さまざまなアプリケーション/API クライアントからのリクエストを処理するには、既存のサービスの API エンドポイントを作成する必要があります。あなたは「デザインファースト」のアプローチに従うことを決定し、OpenAPI 仕様を使用して API の仕様を構築しました。この仕様を使用して、API の実装、インタラクティブな API ドキュメントの生成、API テストケースの生成などを行います。

## Apigee はどのように役立つでしょうか？

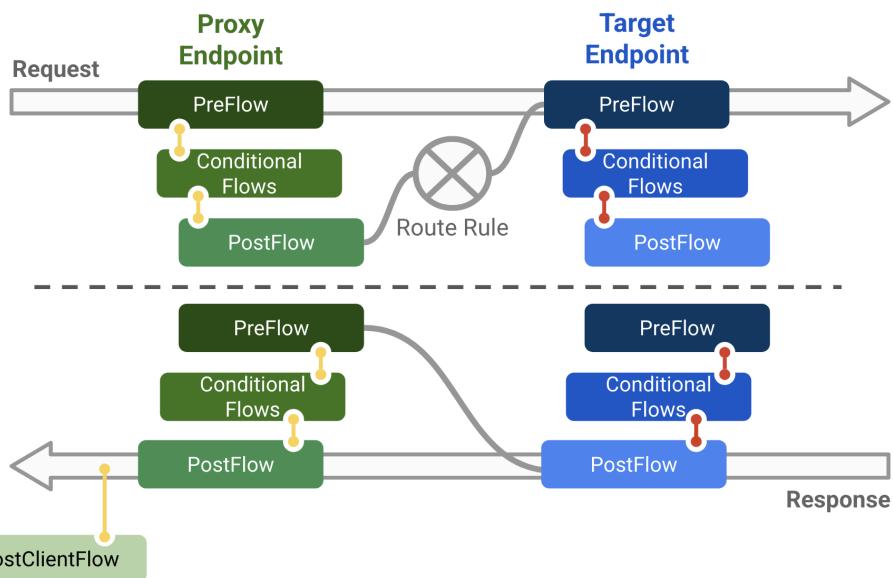
Apigee を使用すると API を迅速に構築して公開できます。これを行うには [API プロキシ](#) を作成します。API プロキシは API エンドポイントとして公開するバックエンド サービスとデータのファサードを提供します。API プロキシは、バックエンド サービスの実装を開発者が使用する API エンドポイントから切り離します。これにより、開発者はサービスに対する将来の変更や実装の複雑さから保護されます。開発者はサービスを更新や変更から分離され、中断されることなく API の呼び出しを続けることができます。Apigee では、API プロキシは API Management 機能のランタイム ポリシー構成が適用される場所でもあります。詳細については、「[API と API プロキシについて](#)」を参照してください。

```

<ProxyEndpoint name="default">
  <PreFlow>
    <Request/>
    <Response/>
  </PreFlow>
  <Flows>
    <Flow name="flow1">
      <Condition/>
      <Request/>
      <Response/>
    </Flow>
  </Flows>
  <PostFlow>
    <Request/>
    <Response/>
  </PostFlow>
  <PostClientFlow/>
  ...
</ProxyEndpoint>

<TargetEndpoint name="default">
  <PreFlow>
    <Request/>
    <Response/>
  </PreFlow>
  <Flows>
    <Flow name="flow2">
      <Condition/>
      <Request/>
      <Response/>
    </Flow>
  </Flows>
  <PostFlow>
    <Request/>
    <Response/>
  </PostFlow>
  ...
</TargetEndpoint>

```

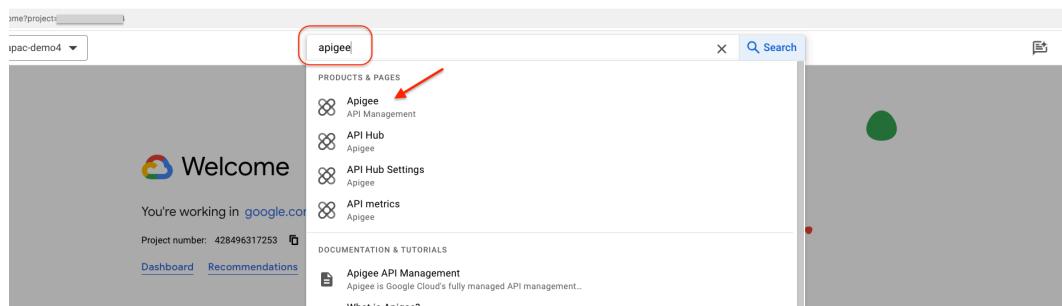


Apigee は、[OpenAPI 仕様](#) を即座に使える様にサポートしているため、API プロキシと関連ドキュメントを自動生成できます。OpenAPI 仕様を設計および作成するには任意のエディタ/IDE を使用できます。選択できるツールは[多数あります](#)。

ラボのこの部分では、受信リクエストを既存の HTTP サービスにルーティングする API プロキシを OpenAPI 仕様から作成する方法を学びます。

## パート 1A - API プロキシの作成

- まず、サンプル API バックエンドの説明として、この [Open API 仕様 YAML ファイル](#) をダウンロードします。このファイルを使用して API プロキシを生成し、後のラボで API を文書化します。
- GCP Console に戻り、検索パネルに [apigee] と入力し、[Apigee API Management] を選択します。



Apigee の概要ページにリダイレクトされます。

- OpenAPI 仕様から Apigee API プロキシを作成するには。サイドナビゲーションメニューから [Proxy development] → [API Proxies] をクリックします。いくつかのプロキシが事前に作成されていることに気づくかもしれません、これからまず新規のプロキシを作成するので、今のところは無視します。

The screenshot shows the Google Cloud Apigee interface. On the left, there's a sidebar with 'Proxy development' expanded, showing 'API proxies' which is also highlighted with a red box. The main area is titled 'API Proxies' and contains a table with two rows: 'hello-world' and 'hipster-mock-api'. A 'CREATE' button is located at the top right of the table area, also highlighted with a red box.

4. [Create] ボタンをクリックします。[Create a proxy] ページで、[Proxy template] ドロップダウンをクリックし、下にスクロールして、OpenAPI spec template 配下の [Reverse proxy (Most common)] を選択します。

The screenshot shows the 'Create a proxy' dialog. On the left, the sidebar includes 'API proxies' under 'Proxy development'. The main dialog has a title 'Create a proxy' and a sub-instruction 'Configure your proxy details below.' Below this, a 'Proxy template \*' dropdown is open, showing options like 'Upload proxy bundle' and 'Integration target'. Under 'OpenAPI spec template', the 'Reverse proxy (Most common)' option is selected and highlighted with a red box. At the bottom of the dialog, there's a URL input field with 'https://', a 'NEXT' button, and a note 'The URL of the backend service that this proxy invokes'. The status bar at the bottom indicates step 2 of 2: 'Deploy (optional)'.

5. 上記の手順 1 でダウンロードした OpenAPI 仕様ファイルを参照します。(ファイル名は hipster-products-backend-spec-v3.yaml である必要があります)。次に、[NEXT] をクリックします。Apigee UI は、OpenAPI 仕様ファイルに記述されている情報を認識し、詳細を自動的に入力することに注意してください。それらを次の値に置き換えます:

- Proxy Name: **Hipster-Products-API**
- Base path: **/v1/hipster-products-api**
- 注: 以下に示すように、必ず **Base path** を手動で更新して、プレフィックス **/v1** を含めてください。

Target (Existing API):

**https://storage.googleapis.com/apigeeexlabs/apijam-lab1/**

The screenshot shows the 'Create a proxy' wizard in the Google Cloud Apigee interface. The left sidebar lists various API management sections like Overview, Proxy development, API proxies, Shared flows, Integrations, Offline debug, API monitoring, Distribution, API products, Portals, Developers, Apps, Monetization, Analytics, API metrics, Custom reports, API security, Risk assessment, and Abuse detection. The main panel is titled 'Create a proxy' and contains the following fields:

- Proxy template \***: Reverse proxy (Most common)
- Proxy name \***: Hipster-Products-API
- Base path**: /v1/hipster-products-api
- Description (Optional)**: Products API for Hipster Application
- Target (Existing API) \***: https://storage.googleapis.com/apigeeexlabs/apijam-lab1/

At the bottom, there are 'NEXT' and 'CREATE' buttons, along with a 'CANCEL' button.

- 重要: 実際にはバックエンドのホスト名または IP アドレスの値をターゲット (既存の API) に入力する必要があります。デモと目的のために、このラボでは GCS バケット上で json ファイルをホストしました。

値を確認し、[NEXT] をクリックします。

6. [Flows (optional)] セクションでは、OpenAPI 仕様で構成されているリストから API 操作/フローを選択できます。すべて選択して [NEXT] をクリックします。

Proxy template \* — Reverse proxy (Most common)

Changing the proxy template will update and reset the form below

- OpenAPI specs
- Proxy details
- ③ Flows (optional)
 

Select the operations to generate conditional flows. You can update the operations later. [Learn more](#)

Path	Verb	Operation	Summary
/products	get	GetProducts	
/products/{productId}	get	GetProductDetails	

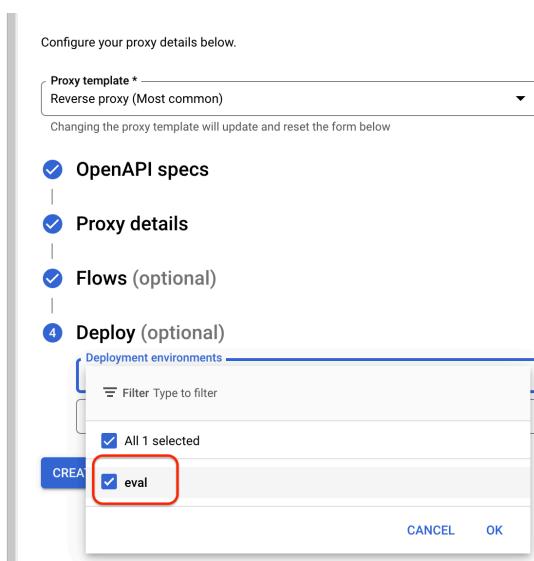
**NEXT**

④ Deploy (optional)

**CREATE**   **CANCEL**

注: 両方のフローを Apigee Proxy にインポートすることを選択しましたが、このラボでは /products の詳細のみを実装します。

7. [Deploy (optional)] セクションで、このプロキシのデプロイ先として eval 環境が選択されていることを確認し、[OK] をクリックしてから [Create] をクリックします。



8. 右下隅で展開の進行状況/ステータスを監視できます。通常、展開には約 30 秒程度かかります。Hipster-Products-API が正常にデプロイされると、次のようなプロキシの詳細ページが表示されます。

The screenshot shows the Google Cloud Apigee interface for the 'Hipster-Products-API'. On the left, a navigation sidebar lists various sections like Overview, API proxies, API products, Portals, Developers, Analytics, API security, and API Hub. The main area is titled 'Proxy summary' under the 'OVERVIEW' tab. It displays a 'Deployments' section with a table showing one revision ('1') in the 'eval' environment. Below it is a 'Revisions' section with a table showing one revision ('1') for the 'Products API for Hipster Application'. A small modal window titled 'Deployment status' is open at the bottom right, showing the message 'Deployed Hipster Products API revision 1 to eval'.

## パート 1B - API プロキシのテストとデバッグ

新しく構築された API プロキシをテストしてみましょう。curl や Postman などの HTTP クライアント、または [Apigee Debug Tool](#)を使用できます。

### 1. APIホスト名を取得する

API をテストする前に、プロキシのホスト名の詳細を取得するには、左側のナビゲーションバーで [Management] → [Environments] に移動します。次に、[ENVIRONMENT GROUPS] をクリックします。[eval-group] 環境グループに割り当てられたホスト名の値をコピーします。

注: 以前にプロビジョニングされた Apigee 組織を使用している場合は、組織内にすでに設定されている環境グループを使用できます。

Name	Host Names	Environments
eval-group	api-qwiklabs-gcp-04-3739922a7f96.apiservices.dev	eval

これが API ホスト名です。このラボでは、このホスト名を使用して API を呼び出します。後で使用できるようにテキストパッドにコピーします。

2. cURL を使用したテスト: cURL または同様のコマンド ライン HTTP クライアントを使用して API をテストするには、次のコマンドを使用します:

```
curl -X GET "https://{{API hostname}}/v1/hipster-products-api/products"
```

すべてがうまくいけば、Hipster Products のリストを JSON 形式で取得できるはずです。

```
student_04_be411fddc174@cloudshell:~ (gwiklabs-gcp-04-3739922a7f96)$ curl -X GET "https://api-gwiklabs-gcp-04-3739922a7f96.api.services.dev/v1/hipster-products-api/products"
{
  "products": [
    {
      "id": "OLJCESPC7Z",
      "name": "Vintage Typewriter",
      "description": "This typewriter looks good in your living room.",
      "picture": "/img/products/typewriter.jpg",
      "priceUsd": 67.98,
      "categories": ["vintage"]
    },
    {
      "id": "66VCHSJNUP",
      "name": "Vintage Camera Lens",
      "description": "You won't have a camera to use it and it probably doesn't work anyway.",
      "picture": "/img/products/camera-lens.jpg",
      "priceUsd": 12.48,
      "categories": ["photography", "vintage"]
    },
    {
      "id": "1YMWNN1N40",
      "name": "Home Barista Kit",
      "description": "Always wanted to brew coffee with Chemex and Aeropress at home?",
      "picture": "/img/products/barista-kit.jpg",
      "priceUsd": 123.99,
      "categories": ["cookware"]
    },
    {
      "id": "L9ECAV7KIM",
      "name": "Terrarium",
      "description": "This terrarium will look great in your white painted living room.",
      "picture": "/img/products/terrarium.jpg",
      "priceUsd": 34.44,
      "categories": ["gardening"]
    },
    {
      "id": "2ZYFJ3GM2N",
      "name": "Film Camera",
      "description": "This camera looks like it's a film camera, but it's actually digital.",
      "picture": "/img/products/film-camera.jpg",
      "priceUsd": 2244.99,
      "categories": ["photography", "vintage"]
    }
  ]
}
```

404 エラーが発生した場合は、以下のトラブルシューティングのヒントを参照してください。それ以外の場合は無視してください。

### トラブルシューティングのヒント

次のヒントを確認してください:

- API プロキシのデプロイが指定された値で正常に完了したかを確認するには、[Proxy Development] → [API Proxies] の [Deployments] セクションに移動します。
- /v1 プレフィックスを含めるようにBase Pathを手動で更新したかを確認するには、Hipster-Products-API プロキシの [Overview] タブ内で、[Revisions] セクションで最新のリビジョンを見つけ、[View] ([Endpoint Summary] の下) をクリックします。

プロキシ エンドポイントのデフォルトのベースパスが

/v1/hipster-products-api で正しく構成されているかどうかを再度チェックします。

- **Base Path** を更新する必要がある場合は、[Develop] タブに戻り、左側のメニューの [Proxy Endpoint] でデフォルトを選択します。XML メインペインを少し下にスクロールし、**HTTPProxyConnection** の下の**Base Path** を更新する必要があります。新しいリビジョンとして保存し、プロキシの新しいリビジョンを再デプロイします。

```

<HTTPProxyConnection>
  <basePath>/v1/hipster-products-api</basePath>
</HTTPProxyConnection>
<!-- Target endpoint -->
<targetEndpoint name="default">
  <!-- Request -->
  <request>
    <!-- Response -->
    <!-- PostFlow -->
  </request>
</targetEndpoint>
<!-- RouteRule -->
<routeRule name="default">
  <!-- TargetEndpoint -->
  <targetEndpoint name="default">/TargetEndpoint</targetEndpoint>
  <!-- RouteRule -->
</routeRule>
</routeRule>

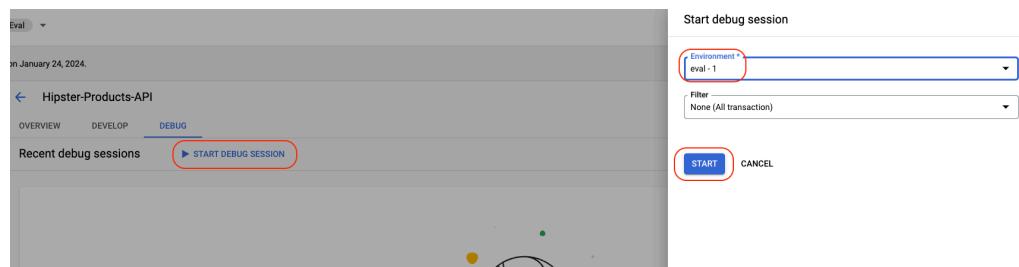
```

ルーティング変更に関する警告が表示された場合は、[Proceed] をクリックします。

### 3. Apigee Debug Toolの使用

エラーをデバッグしたり、潜在的なパフォーマンスのボトルネックを見つけていたりする必要がある場合があります。Apigee には、Apigee Debug Tool と呼ばれる便利なデバッグ/トレース機能が付属しています。この機能には、[ガントチャート](#)で API フローを視覚化できるユーザーインターフェースが付属しています：

- API プロキシの [Debug] タブに移動します。次に、[START DEBUG SESSION] をクリックします。
- デプロイされた API リビジョンが Environment ドロップダウンで選択されていることを確認します。次に、[START] ボタンをクリックします。



- デバッグ セッションが開始されるまで待ちます。
- POSTMANやcurlコマンドなどのツールを使用して、以下のURLにリクエストを送信します：

`https://{{API hostname}}/v1/hipster-products-api/products`

- 組織および環境の API ホスト名を使用するように URL を変更します。また、URL の末尾に `/products` が追加されていることを忘れないでください。
- API プロキシがリクエストを受信し、デバッグ セッションによって記録された HTTP ステータス 200 レスポンスを送り返したことがわかります。リクエスト 200 レコードをクリックして、中央のセクションを応答までスクロールすると、

デバッグ セッションで応答の詳細が表示されます。

The screenshot shows the API Platform interface with the 'Hipster-Products-API' selected. The 'DEBUG' tab is active. On the left, there's a summary of the transaction: ID f74f3a75-a53d-4528-b13d-7a75cf52ee0c, Status In progress, Environment eval, Revision 3, Ends within 8m 9s, and URL ...io/v1/hipster-products-api. Below this is a table of recent transactions, with the first entry (a GET request) highlighted by a red oval.

The main area shows a timeline for the 'GET /v1/hipster-products-api/products' request. The timeline includes events like 'Target Request Flow Started', 'Request Sent', 'Response', 'Response Start', 'Target Response', 'Proxy Response', 'Proxy Response Flow Started', 'Condition', 'CORSResponseOrErrorFlowExecution', and 'Response Sent'. The 'Response Sent' event is highlighted with a red oval.

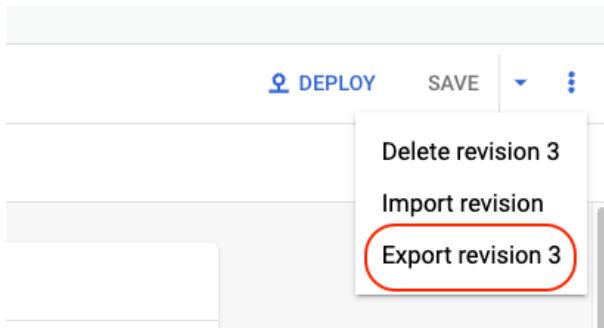
On the right, the 'Response Sent' details are displayed. It shows the start time (@47ms), timestamp (2023-01-27 05:16:09.713 +0800), and various headers: Alt-Svc, content-length, content-type, date, Via, and x-request-id. The 'Response content (3)' section is also shown, with a red oval highlighting the JSON response body:

```
products:[{"id": "OLJCESPC7Z", "name": "Vintage Typewriter", "description": "This typewriter looks good in your living room.", "picture": "/img/products/typewriter.jpg", "priceUsd": 67.98, "category": "Vintage"}, {"id": "66VCHS.JNLP", "name": "Vintage Camera Lens", "description": "You won't have a camera to use it and it probably doesn't work anyway.", "picture": "/img/products/camera-lens.jpg", "priceUsd": 12.48, "categories": ["photography", "Vintage"]}, {"id": "1D7VMW", "name": "Home Barista Kit", "description": "Aawa makes three coffee with the Annex Aeropress at home.", "picture": "/img/products/aeropress-kit.jpg", "priceUsd": 123.99, "categories": ["Cookware"]}, {"id": "L0ECAVJKM", "name": "Terrarium", "description": "This terrarium looks great in your white painted living room.", "picture": "/img/products/terrarium.jpg", "priceUsd": 34.44, "category": "Vintage"}]
```

## パート 1C - API プロキシ (リビジョン) をローカルにエクスポート/インポートする

プロキシに変更を加えるとリビジョンが作成されます。別の機会に再利用する必要がある場合に備えて、API プロキシ (リビジョン) を Zip バンドルとしてローカルに保存しましょう。

1. API プロキシを保存するには、プロキシの [Develop] タブに移動し、[保存] ボタンの横にある [3 つのドット] ボタンの下にある [Export revision] オプションをクリックします。



これで、zip バンドルをローカルに保存できるようになります。

2. 同様に、[Import revision] オプションを選択して、リビジョンをプロキシにインポートできます。

## パート 1 - まとめ

これで、このハンズオン ラボの最初の部分は完了です。ここまでは、OpenAPI 仕様と Apigee プロキシ ウィザードを使用して、Apigee を使用して既存のバックエンドをプロキシする方法を学習しました。また、cURL と Apigee Debug Tool を使用して API プロキシをテストしました。最後に、API プロキシリビジョンをローカルにエクスポートできます。

### 参考文献

#### Apigee ドキュメント

API プロキシに関する役立つ Apigee ドキュメントのリンク:

- [OpenAPI 仕様からの API プロキシの作成](#)
- [API 開発のライフサイクル](#)

### ディスカッションのための質問

1. Deployステップは正確には何のためにあるのでしょうか?プロキシを環境にデプロイせずに作成できますか?
2. API プロキシのベース パスにプレフィックスとして /v1 を配置することは必須ですか?そうしないとどのような影響があるでしょうか?
3. バックエンドなしで API プロキシを作成できますか?
4. パート 1B の Apigee Debug ツールを参照すると、[Filter] というドロップダウンがあることに注目してください。その目的は何ですか?

## パート 2 - Apigee における API セキュリティと API プロデューサー/コンシューマー関係

ペルソナ : API チーム

### ユースケース

さまざまなアプリ (API コンシューマー) で使用できるように、セキュリティを確保して公開したい API があります。API への許可されたアクセスを設定することに加えて、API キーによってアプリを識別し、どのアプリが API を呼び出すことができるかを制御できるようにしたいと考えています。また、呼び出し側アプリに基づいて API 動作をカスタマイズしたり、Apigee Analytics を通じてさまざまなアプリによる API 消費量を測定したりすることができます。

### Apigee はどのように役立つでしょうか？

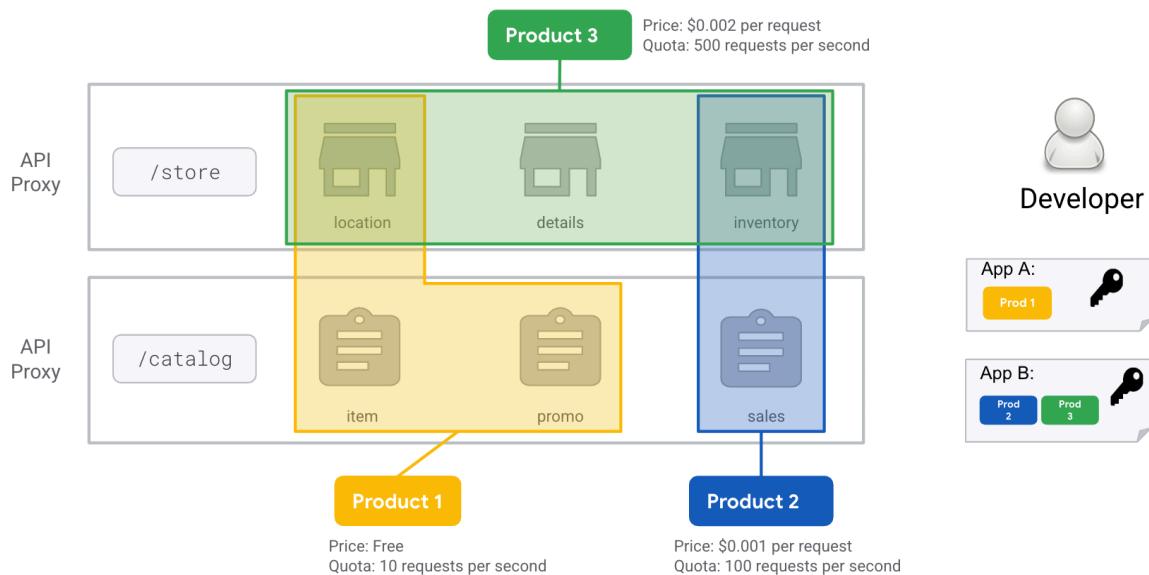
#### API プロキシ - API プロダクト - アプリの関係

Apigee で API プロキシを保護するには、まず API プロキシ、API プロダクト、アプリの関係を理解する必要があります。

API プロキシを使用すると、API 設計仕様に従って API エンドポイントを公開できますが、API バックエンド (ターゲット サービス) をフロントエンド (クライアント アプリ) から分離し、API の生成と消費を分離することもできます。これは、1 つ以上の API プロキシをバンドルし、API を使用する方法を定義する構成である「[API プロダクト](#)」を作成することによって実現されます。API プロダクト構成には、バンドル API の使用に関するルールを定義するメタデータが含まれる場合があります。これらのルールには、許可された消費割り当て (例: 1 分あたり 100 API 呼び出し)、可視性 (パブリック vs プライベート vs 内部)、API リソース制

限 (例: /products リソース URL のみを呼び出すことができ、/products/{product ID} は呼び出すことはできない)、呼び出し元がアクセスを許可される API デプロイメント環境 (例: test、prod) などが含まれる場合があります。

API 製品が作成されると、クライアントアプリはそれらをサブスクライブできます。サブスクリプション時に、Apigee はアプリの API キーとシークレットのペアを自動的に生成してプロビジョニングします。これらの資格情報を使用して、アプリコード内から認証と認可を使用してバンドルされた API リソースを呼び出すことができます。



## APIプロキシ構成

Apigee では、[API キー](#)、[OAuth](#)、[JWT](#)、[SAML](#) など、API を保護し API 呼び出しを承認する複数の方法が提供されていますが、このラボでは、単純な API キー検証を使用して API を保護することに焦点を当てます。

API プロキシ内では、[Verify API Key ポリシー](#) を使用して、API キー検証に基づいて受信 API 呼び出しを認証および承認できます。API キーの検証が成功すると、API キーの検証ポリシーによって、呼び出しを行ったアプリ、アプリ開発者、呼び出しに関連付けられた API 製品などに関する詳細が API プロキシ ランタイム コンテキストにも入力されます。このコンテキストを使用して、適用される他のポリシーをパラメータ化し、クォータの適用やクライア

ント アプリに基づくルーティングなどの API の動作に影響を与えることができます。データを抽出して、Apigee Analytics を通じて分析情報を得るために使用することもできます。

このパートでは次のことを行います:

- 既存の安全でない API プロキシに対して API キーの検証ポリシーを構成し、Apigee Debug Toolを使用してポリシーの動作を確認します。
- API プロキシを API 製品にバンドルします。
- API 製品をサブスクリープする組織内で開発者とアプリを登録し、API の承認された使用をテストします。

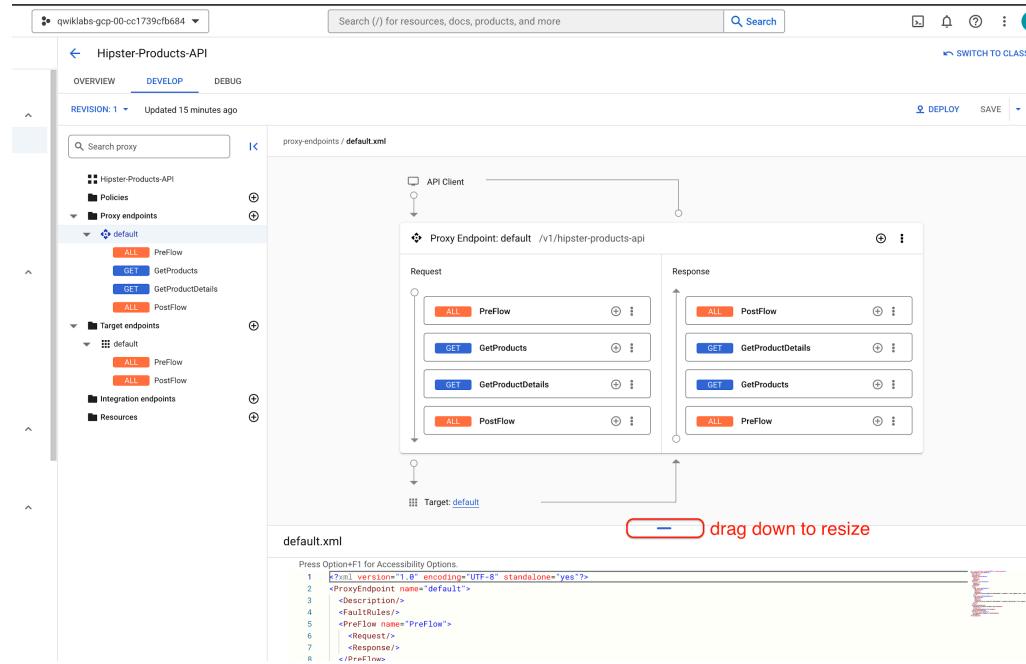
## パート 2A - API キーの検証ポリシーを API プロキシに追加する

- GCP コンソールの Apigee メニューで、最後のパート (プロキシ開発) の API プロキシに移動し、パート 1 で作成した API プロキシ (名前は **Hipster-Products-API** である必要があります) を選択し、プロキシの [Develop] タブをクリックします。

The screenshot shows the Google Cloud Apigee interface. The left sidebar has sections like Overview, Shared flows, Integrations, Offline debug, API monitoring, API products, Portals, Developers, Apps, Monetization, Analytics, API metrics, Custom reports, API security, Risk assessment, Abuse detection, and Security reports. The 'API proxies' section is highlighted with a red box. The main content area shows the 'Hipster-Products-API' proxy. The 'OVERVIEW' tab is active. The 'DEVELOP' tab is selected and highlighted with a red box. The 'DEBUG' tab is also present. Below these tabs, it says 'REVISION: 1' and 'Updated 5 minutes ago'. A search bar 'Search proxy' is present. The 'Proxy endpoints' section shows a 'default' endpoint with methods: ALL, GET, GET, ALL, POST, and flows: PreFlow, GetProducts, GetProductDetails, PostFlow. The 'Target endpoints' section also shows a 'default' endpoint with similar configurations. On the right, there is a large code editor window titled 'Hipster-Products-API.xml' displaying the XML configuration for the proxy.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<APIProxy revision="1" name="Hipster-Products-API">
    <DisplayName/>
    <Description>Products API for Hipster Application</Description>
    <CreatedAt>1700038075774</CreatedAt>
    <LastModifiedAt>1700038075774</LastModifiedAt>
    <BasePaths>/v1/hipster-products-api</BasePaths>
    <ProxyEndpoints>
        <ProxyEndpoint default=</ProxyEndpoint>
        <ProxyEndpoints>
            <TargetEndpoint default=</TargetEndpoint>
            <TargetEndpoints>
                <TargetEndpoint default=</TargetEndpoint>
            </TargetEndpoints>
        </TargetEndpoint default=</TargetEndpoint>
    </ProxyEndpoints>
</APIProxy>
```

2. [Proxy endpoints] ツリー メニューで、**default** をクリックします。UI が API フロー図に変更されるはずです。図全体を表示するには、ペインを下にドラッグしてエディターのサイズを変更するか、ブラウザーをズームアウトする必要がある場合があります。



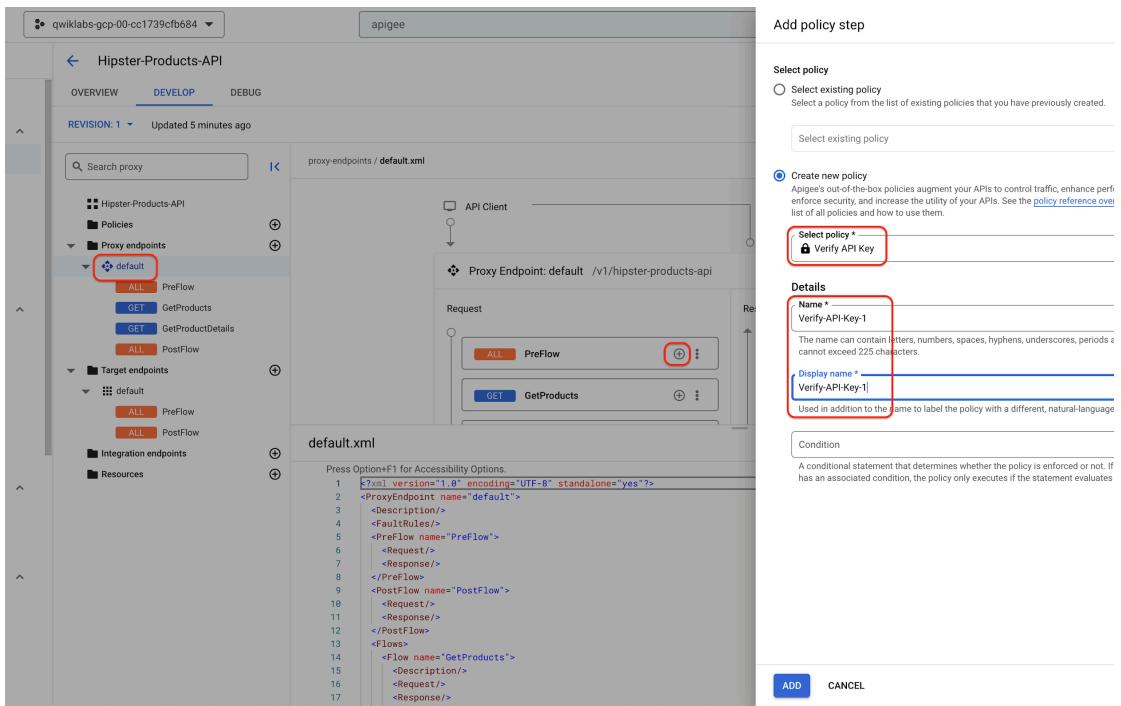
ます。

ここで、[API Client] => [Proxy Endpoint] (Request) => [Target Endpoint] (default) => [Proxy Endpoint] (Response) => API Client のフロー矢印に注目してください。この図は、Apigee の API プロキシによって処理される API クライアントからのフローを視覚化しています。

3. [Proxy endpoint] => **default** ツリー メニューで、リクエストの[Preflow] の(+)をクリックします。[Create new policy] オプションを選択し、[Select policy] ドロップダウンで [Verify API Key] を選択し次の値を指定します：

- Name: **Verify-API-Key-1**
- Display Name: **Verify-API-Key-1**

次に[ADD]をクリックします。



注: [新しい従量課金制\(更新された属性\)の料金モデル](#)を使用してこのラボを実行する場合、[Create new policy]ステップで、[標準ポリシーと拡張ポリシータイプ](#)が表示されるUIにわずかな違いがあることに気づくかもしれません。

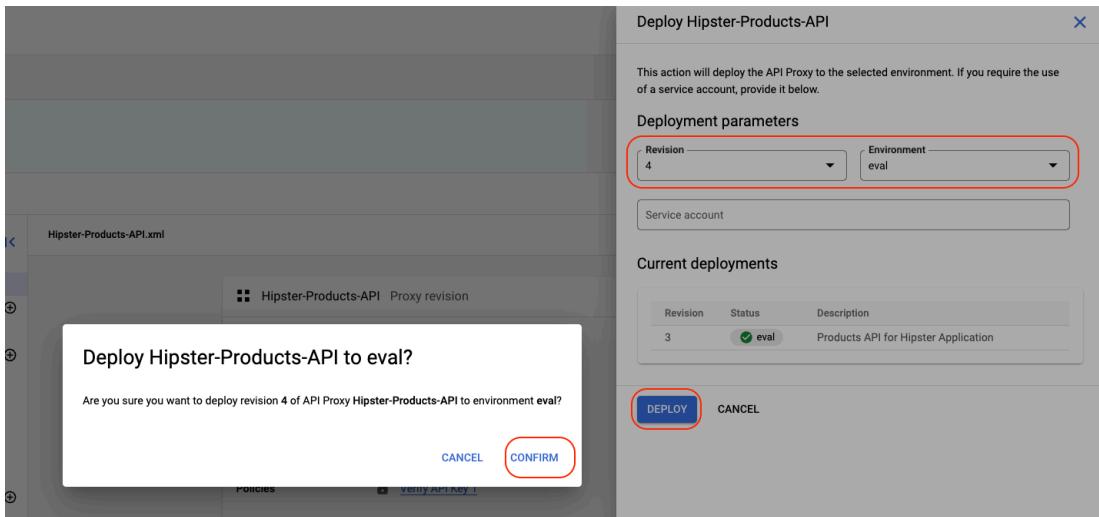
4. Verify API Key ポリシーを選択すると、その構成を確認できます(デフォルトのポリシー構成は以下のとおりです)。確認するAPIキーは、API呼び出しが行われたときに、コンテキスト変数 `request.queryparam.apikey` を使用してAPIプロキシのランタイムコンテキストから取得されることに注意してください。これはデフォルトですが、`request.header.client-id`などの任意のパラメータからキーを取得する

のようにポリシーを構成できます。

The screenshot shows the 'Hipster-Products-API' proxy configuration in the 'DEVELOP' tab. A policy named 'Verify API Key 1' is selected and highlighted with a red circle. The policy is attached to the 'Verify API Key 1' proxy endpoint. The XML code for the policy is displayed in the right panel:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAPIKey continueOnError="false" enabled="true" name="Verify-API-Key-1">
    <DisplayName>Verify API Key 1</DisplayName>
    <Properties/>
    <APIKey ref="request.queryparam.apikey"/>
</VerifyAPIKey>
```

5. API プロキシを保存します。[Save as new revision?] というメッセージが表示された場合は、[SAVE AS NEW REVISION] をクリックして実行します。
6. [DEPLOY] ボタンをクリックし、[Deploy] ポップアップが表示されたら、最新のリビジョンを選択していることを確認し、[Environment] で eval が選択されていることを確認して、[Deploy] をクリックします。次に、確認ダイアログボックスで [CONFIRM] をクリックします。



7. API プロキシの環境へのデプロイには通常、約 30 秒程度かかります。

- 新しいリビジョンが正常にデプロイされたら、プロキシの [Debug] タブをクリックして、セキュリティで保護された API をテストし、[START DEBUG SESSION] をクリックして評価環境を選択し、[Start] をクリックします。
- POSTMANやcURLコマンドなどのツールを使用して、次のURLにリクエストを送信します：

<https://{{API hostname}}/v1/hipster-products-api/products>

Apigee 環境のホスト名を使用するように {{API hostname}} を変更します ([Management] → [Environments] → [Environment Groups]) にあります)。

- 次のようなエラー応答メッセージが表示されるはずです：

```
{"fault":{"faultstring":"Failed to resolve API Key variable
request.queryparam.apikey","detail":{"errorcode":"steps.oauth.v2.FailedToResolveAPIKey"}}}
```

- [Debug] タブに戻ると、以下に示すように、API 呼び出しに対する 401 (未承認) 応答が表示されるはずです。これは、API プロキシがクエリ パラメーターとして API キーを前提としているためです。

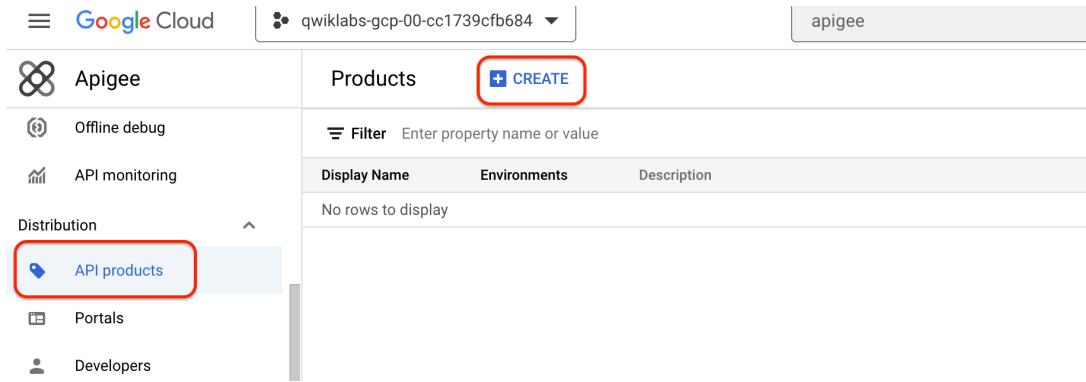
The screenshot shows the Apigee API Debugger interface. On the left, the 'DEBUG' tab is selected. In the center, a transaction table shows a single entry with a status of 401. On the right, the flow timeline shows a 'Verify API Key 1' step where an error occurred. The error details are displayed in a modal: 'Error' (Status: 401), 'Start Time: @:67ms Timestamp: 2023-01-27 (06:31:33.855 +0800)', and 'Variables (0)'.

#	Method	Status	Elapsed
1	GET	401	107ms

次のステップでは、この API 呼び出しを正常に実行できるようにする API キーを取得します。

## パート 2B - API 製品の作成

1. GCP Console で、サイド ナビゲーション メニューから [Distribution] => [API Products] をクリックし、[+ Create] ボタンをクリックします。



2. Product details の下にある次のフィールドに値を入力します。

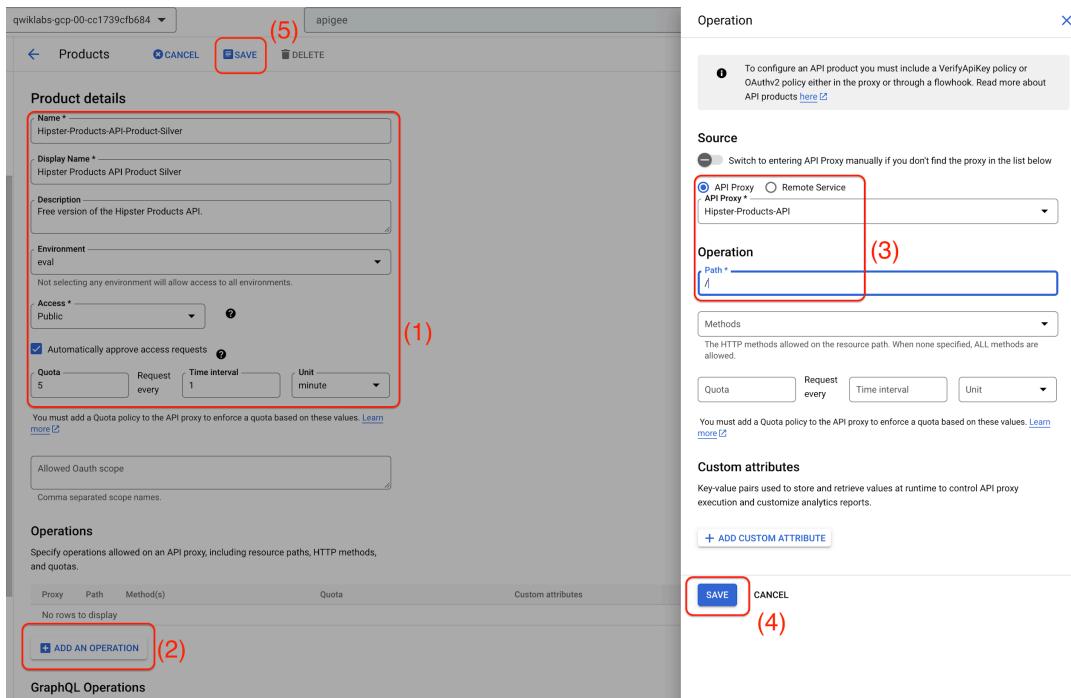
注: 他のユーザーと共有する Apigee 組織を使用している場合は、他のユーザーの作業を誤って変更しないように、API 製品名を含むすべてのアセットの前に \*あなたのイニシャル\* を付けてください。例えば API 製品「名前 = xx\_Hipster-Products-API-Product」

- Name: Hipster-Products-API-Product-Silver
- Display name: Hipster Products API Product Silver
- Description: Free version of the Hipster Products API.
- Environment: eval
- Access: Public
- Quota : 5 requests every 1 minute

注: API 製品には、「Quota」と呼ばれる一連のフィールドがあり、許可する期間ごとのリクエスト数（例：1 [分/時間/日/月]あたり 5 リクエスト）を設定できます。ただし、これを構成するだけではクオータは強制されません。これらのフィールドは、Quota ポリシー (enforcement point) がポリシーを実行するときに動的に参照できるメタデータと考えてください。次のパートでは、

Quota ポリシーを適用した後に新しい効果がどのように適用されるかを説明します。

- [Operation] セクションで、[Add an Operation] をクリックします。
- 画面の右側に [操作] ペインが表示されるので、[API Proxy] ドロップダウンから **Hipster-Products-API** を選択します。
- [Path] に、/ を追加します。  
注: ここでは、API プロキシ全体を API 製品に追加します。プロキシから特定の API リソース パスのみを選択し、それらを API 製品にバンドルすることもできます。
- 操作ペインで [Save] をクリックします。
- API プロダクトを Save します。



## 2C - GCP コンソールで開発者を手動で作成する

ここでの開発者という用語は、API コンシューマーまたは API クライアントを指します。

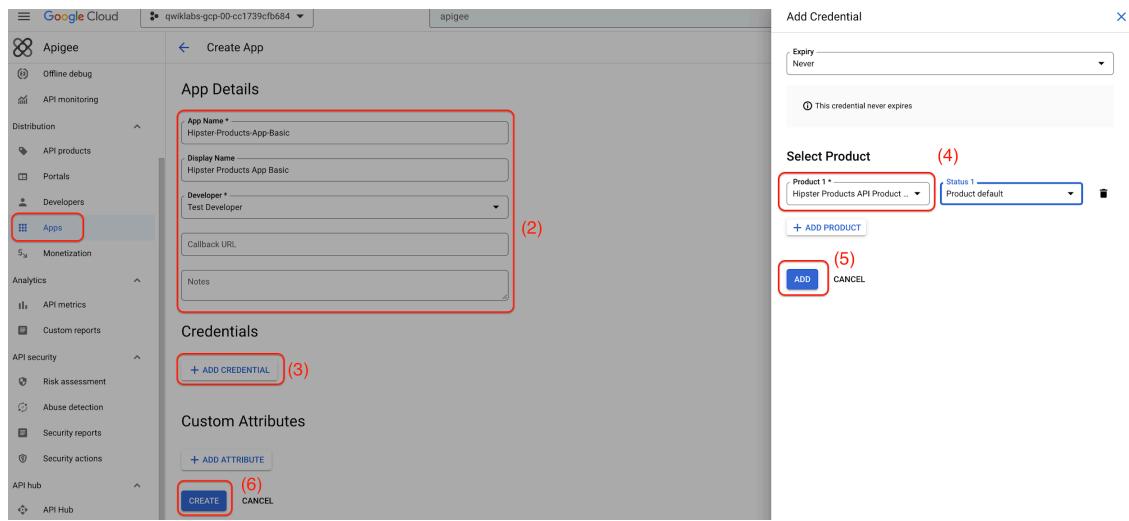
1. サイドナビゲーションメニューの[Distribution] → [Developers] に移動し、[CREATE] ボタンをクリックして開発者を作成します。
2. 次の詳細を入力します:
  - First Name: Test
  - Last Name: Developer
  - Email: testdev@test.com
  - Username: testdev
3. [ADD] をクリックします。

The screenshot shows the Apigee management interface. On the left, there's a sidebar with various options: Offline debug, API monitoring, Distribution (which is expanded), API products, Portals, Developers (which is selected and highlighted in blue), Apps, and Monetization. On the right, the main area is titled 'Add Developer'. It contains four input fields with validation stars: 'First Name \*' with 'Test', 'Last Name \*' with 'Developer', 'Email \*' with 'testdev@test.com' (which is highlighted with a blue border), and 'Username \*' with 'testdev'. At the bottom are two buttons: 'ADD' (in a blue box) and 'CANCEL'.

## 2D - 管理 UI でアプリを手動で登録し、API キーを生成します

1. サイドナビゲーションメニューから[Distribution] → [Apps] に移動し、[CREATE] ボタンをクリックします。
2. 次の詳細を入力します:
  - Name: Hipster-Android-App-Basic
  - Display Name: Hipster Android App Basic
  - 以前に作成した開発者を選択します。開発者が表示されない場合は、ブラウザを更新してください。

3. [Credentials] セクションで、[Add Credential] ボタンをクリックします。
4. 右側に [Add Credential] ペインが表示されるので、以前に作成した **Hipster Products API Silver** を選択します。
5. [Add] をクリックします。
6. [Create] ボタンをクリックして保存します。



7. 作成が完了すると、Apigee がアプリの API キーとシークレットのペアを生成したことがわかります。キーの横にある[Show/Hide] (目のアイコン) リンクをクリックし、この API キーをコピーします。

The screenshot shows the Apigee interface for managing an application. On the left, there's a sidebar with various sections like Offline debug, API monitoring, Distribution, Monetization, Analytics, API security, API hub, and Management. The 'Apps' section is currently selected. The main right panel shows 'App Details' for an app named 'Hipster-Products-App-Basic'. It lists the status as Approved, name as Hipster Products App Basic, registered date as November 15, 2023, developer as Test Developer (testdev@test.com), and callback URL. Below this is a 'Notes' section. A large modal window titled 'Credential' is open, showing a status of Approved. It contains a 'Key' field with a value 'Ejb0ZiGXEcZIHfZvzVQMHkmlusLnN11RNySO2U25l4An1jv2' (which is highlighted with a red box) and a 'Secret' field with a masked value. The 'Expiry' field is set to 'Never'. At the bottom of the modal, there's a table with one row showing 'Hipster-Products-API-Product-Silver' and 'Status Approved'. The entire 'Credential' modal is also highlighted with a red box.

Apigee

View App EDIT DELETE

Offline debug

API monitoring

Distribution

API products

Portals

Developers

Apps

Monetization

Analytics

API metrics

Custom reports

API security

Risk assessment

Abuse detection

Security reports

Security actions

API hub

API Hub

API Hub Settings

Management

App Details

Status Approved

Name Hipster-Products-App-Basic

Display name Hipster Products App Basic

Registered November 15, 2023

Developer Test Developer (testdev@test.com)

Callback URL

Notes

Credentials

Credential

Status Approved

Key Ejb0ZiGXEcZIHfZvzVQMHkmlusLnN11RNySO2U25l4An1jv2

Secret

Expiry Never

Products	Status
Hipster-Products-API-Product-Silver	Approved

これで、この API キーを使用して、API プロキシに対して有効な API リクエストを行うことができます。

## 2E - 有効な API キーを使用して API をテストする

1. API プロキシの [Debug] タブに移動します。
2. [START DEBUG SESSION] をクリックします。
3. デプロイされた API リビジョンが [Environment] で選択されていることを確認します。

4. [START] ボタンをクリックします。
  - デバッグ セッションが開始されるまで待ちます。
5. POSTMANやcURLコマンドなどのツールを使用して、以下のURLにリクエストを送信します:

`https://{{API hostname}}/v1/hipster-products-api/products?apikey={{API Key}}`

上記URLでは:

- {{API hostname}} は、以前に[Management] >> [Environments] >> [Environment Groups] >> あなたの環境グループからコピーした環境グループのホスト名です。
  - {{API Key}} は、新しく作成したアプリ構成からコピーした API キーです。
6. Verify API Key ポリシーによって API キーが有効であることが検出されたため、API リクエストが 200 OK 応答を返していることがわかります。

The screenshot shows the Apigee Edge interface for the 'Hipster-Products-API'. On the left, under the 'OVERVIEW' tab, there is a table of transactions. One transaction is highlighted with a red box around the 'Method' column, which shows 'GET'. To the right of the table, a detailed view of a 'Debug session' is shown. This view includes a timeline chart and a policy flow diagram. The timeline chart shows a single transaction taking 114 ms. The policy flow diagram shows a sequence of events: 'Verify API Key 1' (Condition), 'Target Request', 'Target Request Flow Started', 'Request Sent', 'Response', 'Response Start', 'Target Response', 'Target Response Flow Started', 'Proxy Response', and 'Proxy Response Flow Started'. A red box highlights the 'Verify API Key 1' condition step. On the far right, a table titled 'Verify API Key 1' lists various configuration details such as operation type, client ID, and proxy paths.

ID	Method	Status	Elapsed
acea3d38-bc13-4fb8-acf4-5e309749bc08	GET	200	155ms

Name	Policy type	Elapsed
Verify API Key 1	oauthV2	114 ms

graphql.operation.operationType	
mint.mining_is_apiproduct_monetized	false
oauthV2.Verify-API-Key-1	/products
request.queryparam.apikey	0XnW7oEHFuxuKA3qRqh1qAvwH1NUmdWkVtGWAuprkF2idTW
request.verb	GET
verifyapikey.Verify-API-Key-1.apiproduct	public
verifyapikey.Verify-API-Key-1.apiproduct	Hipster-Products-API-Product
verifyapikey.Verify-API-Key-1.apiproduct	/
verifyapikey.Verify-API-Key-1.apiproduct	OPERATION
verifyapikey.Verify-API-Key-1.client_id	0XnW7oEHFuxuKA3qRqh1qAvwH1N
verifyapikey.Verify-API-Key-1.client_s...	sLclcq16svXkwGbVopVy6RXOIXYoI
verifyapikey.Verify-API-Key-1.develo...	1d154053-6798-4b4a-89ed-b55d6b0
verifyapikey.Verify-API-Key-1.develo...	Hipster-Products-App
verifyapikey.Verify-API-Key-1.develo...	welylau@google.com
verifyapikey.Verify-API-Key-1.develo...	apigee-lab2@{@72609c86-49db-4d
verifyapikey.Verify-API-Key-1.Display...	Hipster-Products-App
verifyapikey.Verify-API-Key-1.expires...	0
verifyapikey.Verify-API-Key-1.failed	false
verifyapikey.Verify-API-Key-1.issued...	1674802894506

## パート 2 - まとめ

ラボのこの部分では、API プロキシ、API 製品、およびアプリの間の関係が、API の生成と API の消費をどのようにわかりにくくするのに役立つかを学びました。また、Verify API Key ポリシーを使用して API プロキシを保護する方法についても説明します。次に、ポリシーを実装し、組み込みのデバッグ ツールを使用してテストしました。

## 参考文献

### Apigee ドキュメント

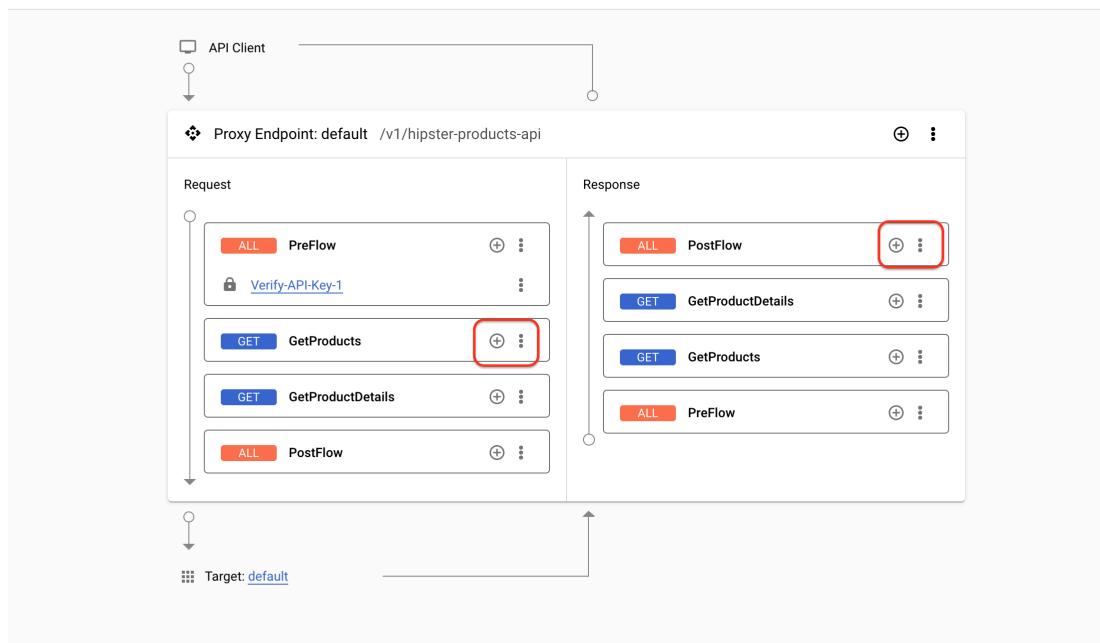
- [VerifyAPIKey ポリシー](#)
- [API プロダクトの概要](#)
- [アプリ デベロッパーの登録](#)
- [アプリ登録を使用した API へのアクセスの管理](#)

## ディスカッションのための質問

1. Proxy EndpointとTarget Endpointがあるのはなぜですか?それが役立つシナリオは何か思い描きますか?
2. 複数のProxy EndpointとTarget Endpointを使用できますか?
3. ラボでは、Proxy Endpointのリクエスト Preflowに Verify API Key ポリシーを追加しました。Proxy Endpointのリクエスト GetProducts フロー や プロキシ エンドポイントのレスポンス Postflowなどの別の場所、あるいはTarget Endpointのどこかに

配置できますか?影響は何ですか?

iroxy-endpoints / default.xml



# パート 3 - API 呼び出しクオータと動的応答を通じた段階的な API 製品サブスクリプション管理

ペルソナ：API プロダクトチーム & API 開発チーム

## ユースケース

呼び出しクオータ: API 階層化は製品としての API という新しい視点を提供します。階層化では、基本オプションとしてのレベル (シルバーなど) を提供します。この製品オファーは、プレミアム API 製品へのアップセルの可能性を伴うエントリーポイントとなります。目標は追加の機能レベルにアップセルすることです。この用語は「フリーミアム」とも呼ばれます。基本的なアプローチは次のとおりです。基礎的な機能または通信クオータをエントリーレベルとして提供し、より多くのデータ アクセスや機能が必要な場合は、これらのオプションを有料で提供します。これにより、開発者は十分な情報に基づいて購入を決定する前に、実際に動作するプロトタイプを作成し、実際のシナリオで API を検討する機会が得られます。

サブスクリプション層に基づく異なる応答結果: プレミアムサブスクライバーがすべて (完全な) 結果を取得し、ベーシックサブスクライバーは結果のサブセットのみを取得するユースケースに遭遇したことがありますか？

## Apigee はどのように役立つでしょうか？

Apigee は、API プロキシの機能ロジックから抽象化された API 製品の概念を提供します。プロキシは、API 製品に定義されている制限にアクセスして適用できます。このようにして、API 製品チームはビジネス モデル (例: API 製品ごとに使用量割り当てや資格のある API オペレーションの確立) に集中できる一方で、API 開発チームはこれらの値を使用してパラメータ化された動作を実装します。

avascript などの拡張ポリシーを利用して追加のロジックを挿入し、Basic サブスクライバの結果のサブセットを削除できます。

## 3A - Quota ポリシーの作成と構成

前述のとおり、クオータは API プロキシにQuota ポリシーを追加することによってのみ適用されます。API プロダクトの Quota フィールドを構成すると、有効な API キーを提示する API 呼び出しが行われると、Apigee は関連付けられた API プロダクトのメタデータ (Quota フィールドを含む) を自動的に取得し、メタデータは Quota (またはその他の) ポリシーから動的に参照できるようになります。

1. サイド ナビゲーション メニューから [Proxy Development] → [API Proxies] に移動し、以前に作成した既存の API プロキシを開きます。
2. プロキシの [Develop] タブに移動し、[Verify-API-Key-1] のポリシーが適切なポリシー名で存在することを確認します。ポリシーをクリックし、以下の XML 構成でポリシー名を確認します。

The screenshot shows the Apigee API Proxy Development interface for the 'Hipster-Products-API'. The left sidebar shows the proxy structure with sections for Policies, Proxy endpoints, Target endpoints, and Integration endpoints. The 'Policies' section is expanded, and the 'Verify API Key 1' policy is selected and highlighted with a red oval. The right panel displays the XML configuration for this policy:

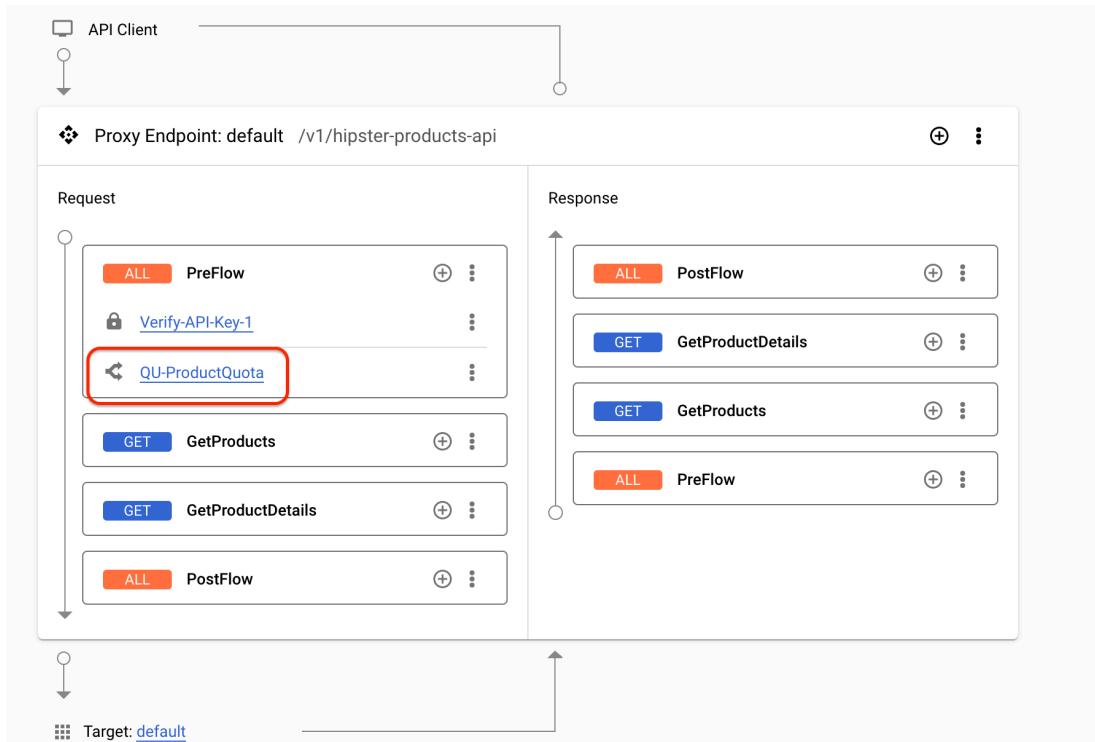
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAPIKey continueOnError="false" enabled="true" name="Verify-API-Key-1">
    <DisplayName>Verify API Key 1</DisplayName>
    <Properties/>
    <APIKey ref="request.queryparam.apikey"/>
</VerifyAPIKey>
```

3. 引き続き、[Proxy endpoint] => [default] ツリー メニューで、Request Preflow の (+) をクリックします (Verify-API-Key-1 も表示されます)。[新しいポリシーの作成] オプションを選択し、[Select policy] ドロップダウンで [Quota] ポリシーを選択します。NameとDisplay nameの両方に QU-ProductQuota の名前を指定

します。次に [ADD] をクリックします。

The screenshot shows the Apigee API Designer interface for the 'Hipster-Products-API'. On the left, the proxy structure is visible with 'Proxy endpoints' expanded, showing 'default' selected. In the main workspace, under 'Request', there is a 'PreFlow' step containing a 'Verify-API-Key-1' policy and a 'GET GetProducts' step. A modal window titled 'Add policy step' is open on the right, showing the 'Create new policy' section selected. A policy named 'QU-ProductQuota' is being created, with its 'Name' set to 'QU-ProductQuota' and 'Display name' set to 'QU-ProductQuota'. The 'Condition' field is empty. At the bottom of the modal, the 'ADD' button is highlighted with a red box.

4. 追加すると、QU-ProductQuota ポリシーが Verify-API-Key-1 ポリシーのすぐ下に表示されます。



5. プロキシでは、前の手順で構成した VerifyAPIKey ポリシーを使用して、有効な API キーを検証した後、API 呼び出し中にプロキシのランタイム コンテキストに次の変数を設定する必要があります。

- `verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.limit`
- `verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.interval`
- `verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.timeunit`

6. 次に、これらの変数を使用してQuota ポリシーをパラメータ化し、キーが対応する API 製品に基づいて動的クオータを適用します。

QU-ProductQuota ポリシーをクリックします。

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota async="false" continueOnError="false" enabled="true" name="QU-ProductQuota">
    <DisplayName>QU-ProductQuota</DisplayName>
    <Allow count="3" />
    <countRef>verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.limit</countRef>
    <Interval ref="verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.interval">1</Interval>
    <TimeUnit ref="request.header.quota_timeout">month</TimeUnit>
    <StartTime>2013-08-21 10:00:00</StartTime>
    <AsynchronousConfiguration>
        <SyncIntervalInSeconds>20</SyncIntervalInSeconds>
        <SyncMessageCount>5</SyncMessageCount>
    </AsynchronousConfiguration>
</Quota>

```

QU-ProductQuota.xml の内容を次のものに置き換えます:

```

Unset
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota async="false" continueOnError="false" enabled="true" name="QU-ProductQuota">
    <DisplayName>QU-ProductQuota</DisplayName>
    <Allow count="3" />
    <countRef>verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.limit</countRef>
    <Interval ref="verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.interval">1</Interval>

```

```
<TimeUnit  
ref="verifyapikey.Verify-API-Key-1.apiproduct.developer.quota.timeunit">minu  
te</TimeUnit>  
    <Distributed>true</Distributed>  
    <Synchronous>true</Synchronous>  
</Quota>
```

上記の Quota ポリシー設定では、フィールドが API 製品に設定されていない場合、デフォルトで 1 分あたり 3 回の呼び出しが許可されます。

7. [SAVE] をクリックし、[SAVE AS NEW REVISION] をクリックして、[Deploy] をクリックします。

注: ポリシーの実行の結果として設定されるコンテキスト変数には、「.(ドット)」で接続された変数名の一部としてポリシーの名前 (プロキシで定義されているとおり) が含まれます。たとえば前の手順で、API キーの検証ポリシーに「Check-API-Key」という名前を付けた場合、次の名前のランタイム コンテキスト変数に「limit」変数 (キーに関連付けられた API プロダクトメタデータから派生) が見つかります:  
`verifyapikey.Check-API-Key.apiproduct.developer.quota.limit``

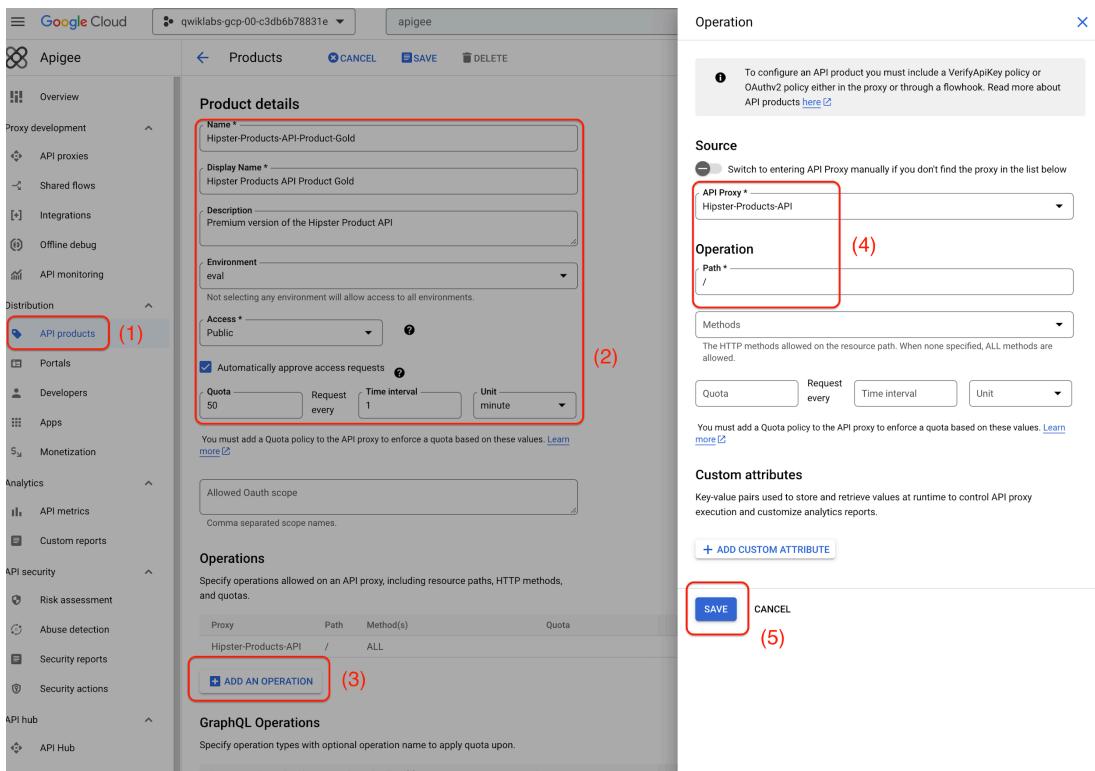
## 3B - ゴールド API プロダクトの作成

パート 2 でシルバー API プロダクトを作成しました。このセクションでは、同じ API プロキシをバンドルするが、異なるクオータが関連付けられた別の API 製品 (ゴールドレベル) を作成します。ゴールド層はプレミアム加入者 (開発者) を対象としています。

1. GCP Console で、[Distribution] → [API Products] に移動します。
2. [Create] をクリックします。
3. 次のフィールドに値を入力します
  - Product details

- Name: **Hipster-Products-API-Product-Gold**
- Display name: **Hipster Products API Product Gold**
- Description: **Premium version of the Hipster Product API**
- Environment: **eval**
- Access: **Public**
- Quota: **50 requests every 1 minute**
- 注: API プロダクトには、**Quota** と呼ばれる一連のフィールドがあり、許可する期間ごとのリクエスト数 (例: 1 [分/時間/日/月]あたり 5 リクエスト) を構成できます。ただし、これを構成するだけではクォータは強制されません。これらのフィールドは、**Quota ポリシー** (施行ポイント) がポリシーを施行するときに動的に参照できるメタデータと考えてください。
- [Operation] セクションで、[Add an Operation] をクリックします。
- 画面の右側に [操作] ペインが表示されるので、[API プロキシ] ドロップダウンから **Hipster-Products-API** を選択します。
- [Path] に、/ を追加します。
- 注: ここでは、API プロキシ全体を API 製品に追加します。プロキシから特定の API リソース パスのみを選択して、それらを API 製品にバンドルすることもできます。

- 操作ペインで [Save] をクリックします。(ただし、まだ API プロダクトの [SAVE] をクリックしないでください)



- API プロダクトを Save します。([SAVE] ボタンを表示するには、少し上にスクロールします)
- [Distribution] => [API Products] に移動すると、2 つの API 製品が存在するはずです。

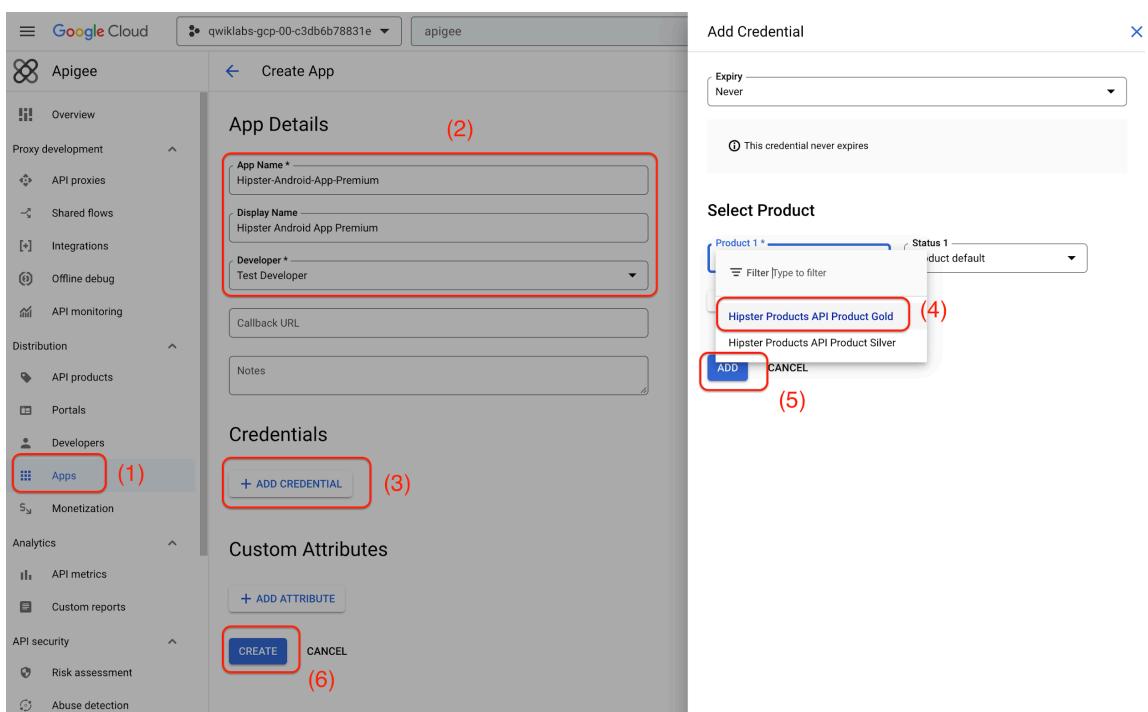
Google Cloud		Products	<a href="#">CREATE</a>			
		Display Name	Environments	Description	Modified	Actions
		Hipster Products API Product Gold	eval	Premium version of the Hipster Product API	2 minutes ago	
		Hipster Products API Product Silver	eval	Free version of the Hipster Products API.	39 minutes ago	

## 3C - ゴールド層 API 製品のプレミアム アプリを作成する

- [Distribution] → [Apps] を選択し、[CREATE] をクリックします。
- 次のフィールドに値を入力します

- App Details
  - Name: **Hipster-Android-App-Premium**
  - Display name: **Hipster Android App Premium**
  - Developer: 既存の任意のDeveloperを選択

3. [Credentials] セクションで、[Add Credential] ボタンをクリックします。
4. 右側に [Add Credential] ペインが表示されるので、以前に作成した Hipster Products API Gold を選択します。
5. [Add Credential] ペインで [Add] をクリックします。



6. [CREATE] をクリックしてこのアプリを作成します。
7. [Distribution] → [Apps] に戻ると、2 つのアプリ (Hipster Android App Premium と Hipster Android App Basic) が表示されます。

Apigee	Apps	CREATE			
Developers	Filter	Filter Apps			
Monetization	Status	App name	Developer name	Registered since	Actions
	Approved	Hipster Android App Premium	Test Developer	November 15, 2023	⋮
	Approved	Hipster Products App Basic	Test Developer	November 15, 2023	⋮

## 3D - さまざまなアプリのクオータをテストする

これで、[Product-App] のペアが 2 つあり、API 製品でクオータを動的に構成しました。それらをテストしてみましょう。

1. [Distribution] → [Apps] メニューで、まず [Hipster Android App Basic] をクリックし、Key (Secretではありません) をコピーします。次に、それを VS Code やメモ帳などのエディターに貼り付けます。

The screenshot shows the API Management interface. On the left, there's a sidebar with various sections like Overview, Proxy development, Distribution, Analytics, and Advanced API Security. The 'Apps' section is currently selected. In the main area, there's a 'App Details' panel and a 'Credentials' panel.

**App Details**

Status	Approved
Name	Hipster-Android-App-Basic
Display name	Hipster Android App Basic
Registered	January 12, 2024
Developer	<a href="#">Test Developer (testdev@test.com)</a>
Callback URL	
Notes	

**Credentials**

**Credential**

Status	Approved
Key	..... <a href="#">Copy</a> (The 'Copy' button is highlighted with a red box)
Secret	..... <a href="#">Copy</a>
Expiry	Never

**Products**

Products	Status
<a href="#">Hipster-Products-API-Product-Silver</a>	Approved

2. ここで、次のようなツールを使用して、次の URL に対して少なくとも 7 回連続リクエストを行ってみましょう:

POSTMAN or cURL command:

`https://{{API hostname}}/v1/hipster-products-api/products?apikey={{API Key}}`

上記URLでは:

- {{API hostname}} は、以前に [Admin] >> [Environments] >> [Groups] >> あなたのenvironment groupからコピーした環境グループのホスト名です。
  - {{API Key}} は、Silver用に新しく作成したアプリ、つまり **Hipster Android App Basic** からコピーした API キーです。
3. うまくいけば、最初の 5 回までは有効な結果が得られるはずです。ただし、6 回目以降はエラーが発生するはずです。

```
{"fault": {"faultstring": "Rate limit quota violation. Quota limit exceeded."}, "Identifier": "_default", "detail": {"errorcode": "policies.ratelimit.QuotaViolation"}}
```

これは、1 分あたり 5 回の呼び出しを定義したためです。

4. 別の呼び出しを行う前にさらに 1 分程度待った場合は、次の 1 分間のクオータがリセットされるため、成功した結果が得られるはずです。
5. 別のアプリ (Hipster Android App Premium) についても同じことを繰り返します。5回以上呼び出せますか？あなたならできるはずです！実際、ゴールド製品では 1 分あたり 50 コールに設定しました。

## 3E - ティアに基づいて応答を動的に変更する

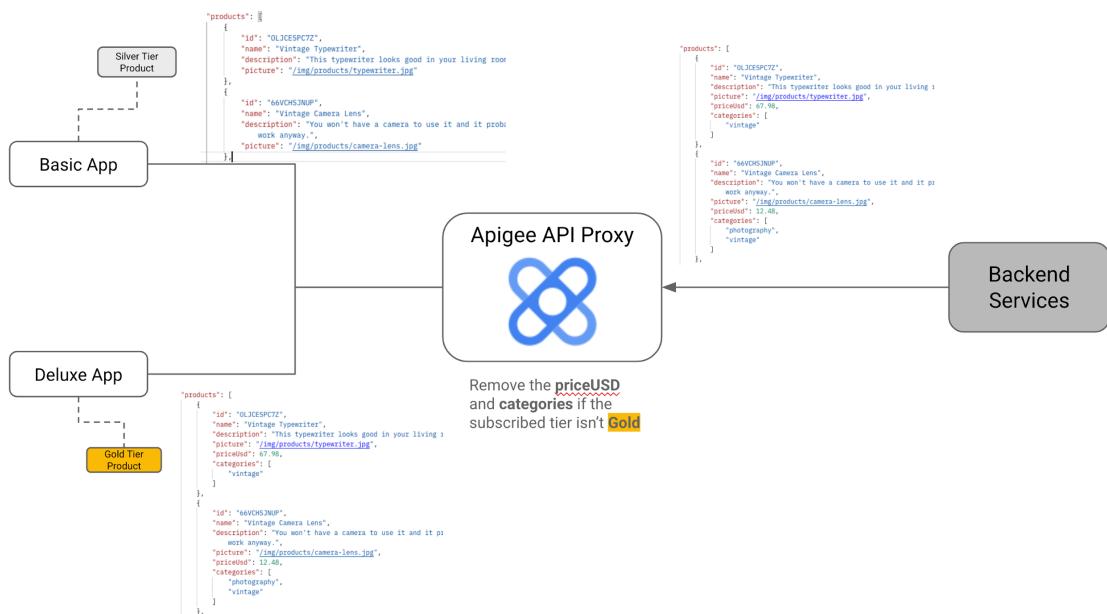
シルバーの API コンシューマー (ベーシックアプリ) はフィールドのサブセットのみを取得するのに対し、ゴールドの API コンシューマー (プレミアムアプリ) はフィールドの完全なリストを取得するというシナリオに遭遇したことがありますか？

通常、2つの異なる API プロキシを使用してこれを実現できますが、追加のオーバーヘッドに対処する必要があります。

Apigee では、Apigee プロキシ、JavaScript ポリシー、API 製品のカスタム属性を組み合わせて、非常にエレガントな方法でこれを実行できます。

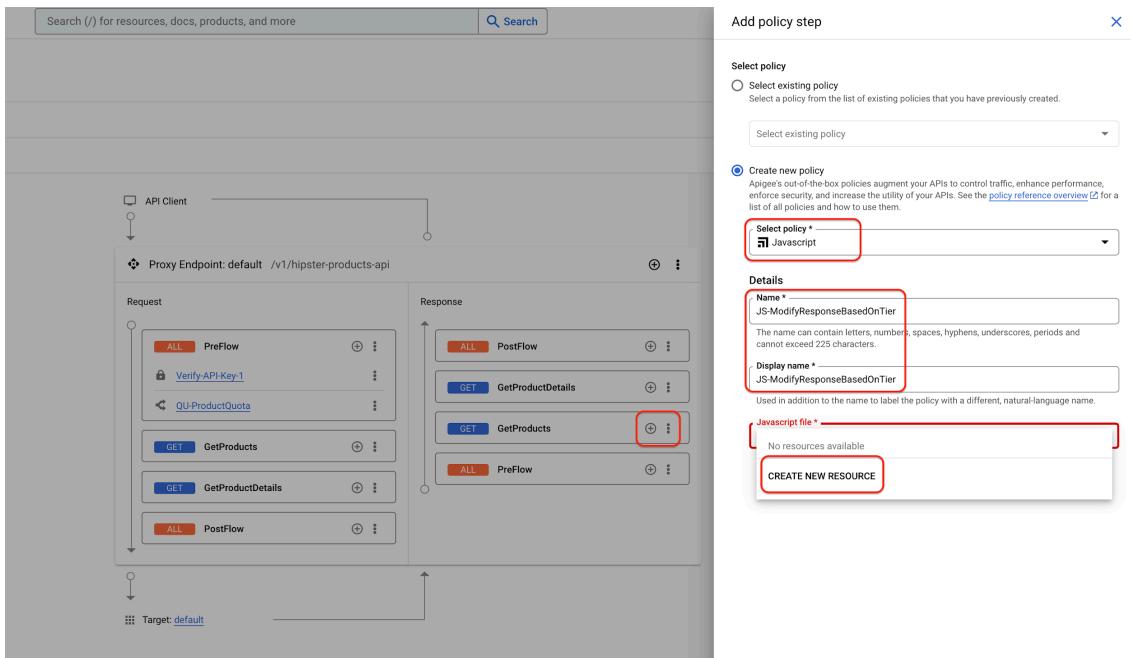
その方法を学びましょう!

1. Hipster Products API の呼び出しに成功したときに得られたレスポンスを思い出してください。上記のパート 1B ステップ 2 を参照してください。 **ID**、**name**、**description**、**picture**、**priceUsd**、**categories** の属性を持つ製品のリストが表示されます。このセクションで行うことは、サブスクライブされたアプリがゴールドではない場合(つまりシルバー)に、属性のうち 2 つを削除することです。次の図はそのアイデアを示しています。



2. 図が中央に正しく表示されるように、[Proxy Endpoint] で [default] をクリックしていることを確認してください。(Requestではなく) Response 内の [GetProducts

の(+)] をクリックします。



3. [Add policy] ステップ ペインが右側に表示されたら、[Create new policy] オプションを選択し、[Select policy] ドロップダウンで Javascript を参照します。名前と表示名の両方に [JS-ModifyResponseBasedOnTier] と入力します。
4. [JavaScript file] フィールドで、[CREATE NEW RESOURCE] を選択します。  
[Add resource] ペインが表示されたらすぐに、[Resource type] で [Javascript] が選択されていることを確認し、[Source] で新しいファイルを作成します。リソース名はそのままにしても構いません (Resource-1.js)。次に [ADD] をク

リックします。

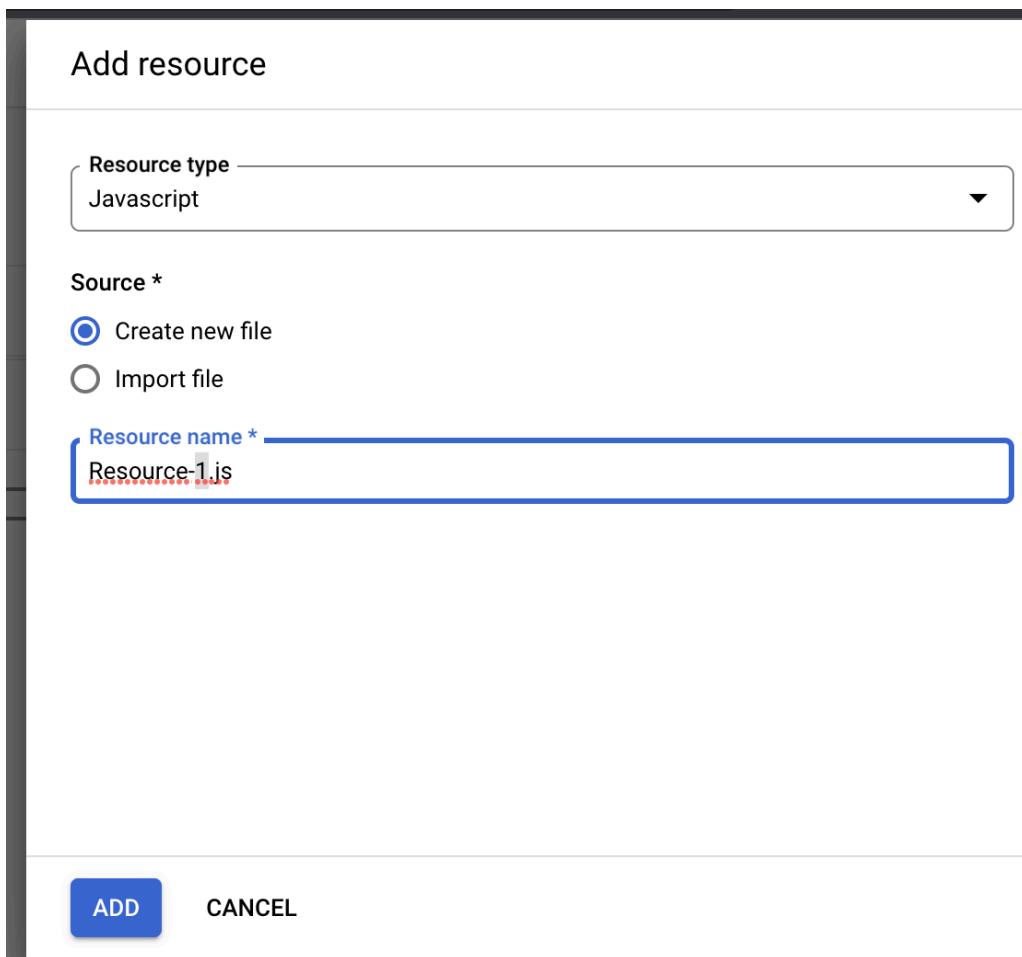
Add resource

Resource type  
Javascript

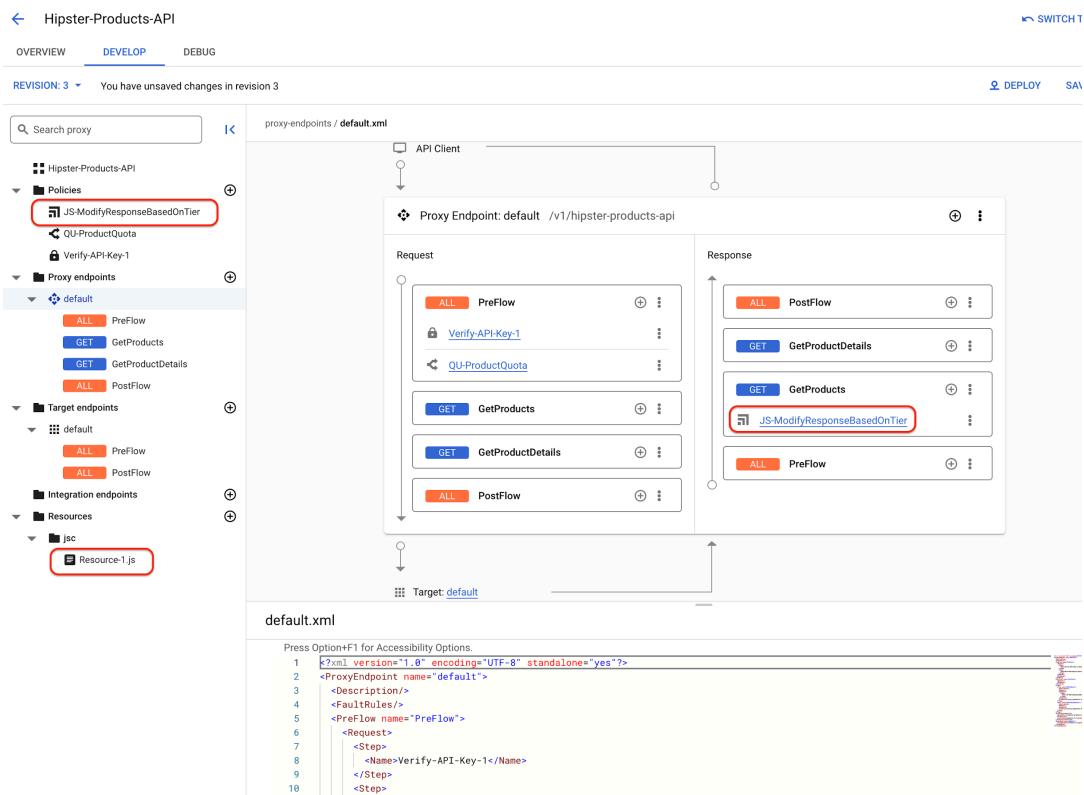
Source \*  
 Create new file  
 Import file

Resource name \*  
Resource-1.js

**ADD** CANCEL



5. [Add policy] ステップペインに戻るはずです。[JavaScript file] フィールドで、**Resource-1.js** ファイルが選択されていることを確認します。次に [ADD] をクリックします。
6. エディターのダイアグラムは次のようになります。2つの項目が追加されていることに注意してください。Policies の下の JS-ModifyResponseBasedOnTier と Resources/jsc の下の Resource-1.js です。



7. **Resource-1.js** ファイル (左側のペインのツリー メニュー上) をクリックし、次のコード スニペットを Resource-1.js コード エディター ペインにコピーします。

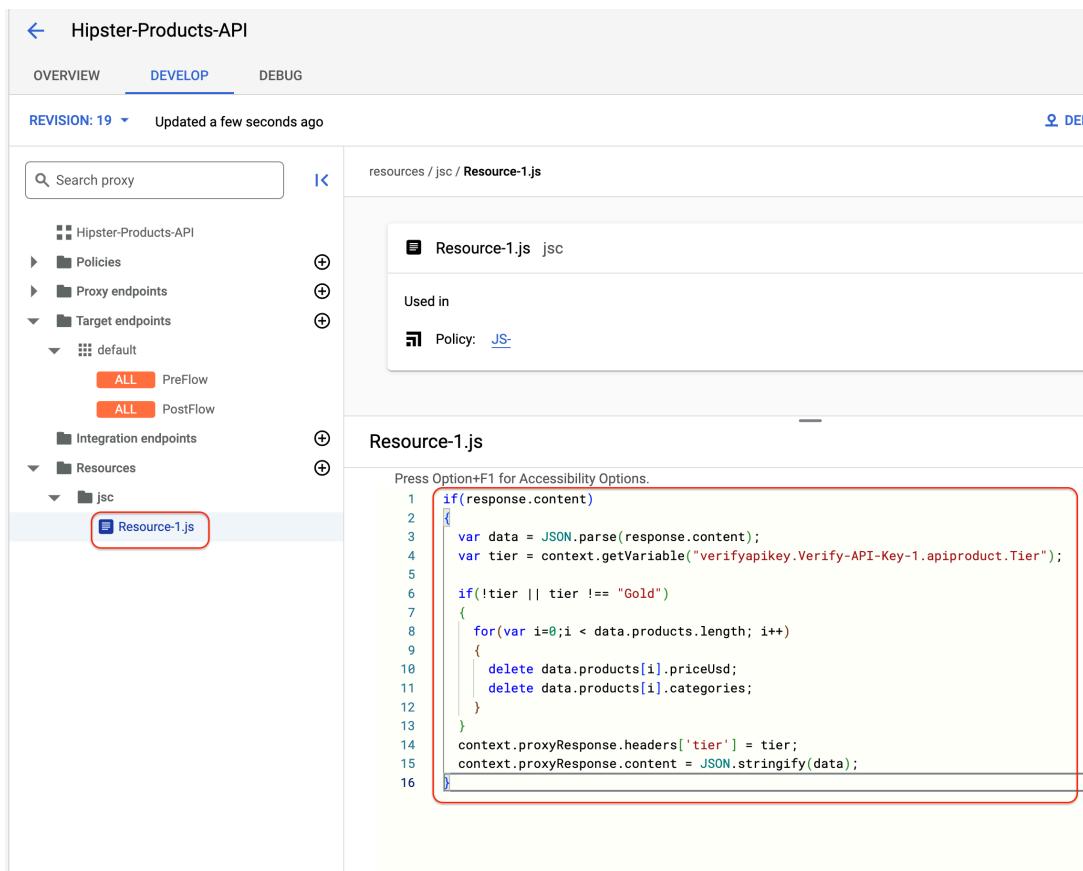
```

JavaScript
if(response.content)
{
    var data = JSON.parse(response.content);
    var tier =
context.getVariable("verifyapikey.Verify-API-Key-1.apiproduct.Tier");

    if(!tier || tier !== "Gold")
    {
        for(var i=0;i < data.products.length; i++)
        {
            delete data.products[i].priceUsd;
            delete data.products[i].categories;
        }
    }
    context.proxyResponse.headers['tier'] = tier;
    context.proxyResponse.headers['content-type'] = 'application/json';
    context.proxyResponse.content = JSON.stringify(data, null, 2);
}

```

JavaScript ポリシーで [JavaScript オブジェクトモデル](#)を使用して、何らかの操作を実行します。上記のコードはティアがGoldでない場合、またはカスタム属性ティアが設定されていない場合に、各Hipster ProductのpriceUsdとcategoriesを削除します。変更されたresponseのコンテンツを出力します。さらに、Tier と呼ばれる responseヘッダーも追加します。



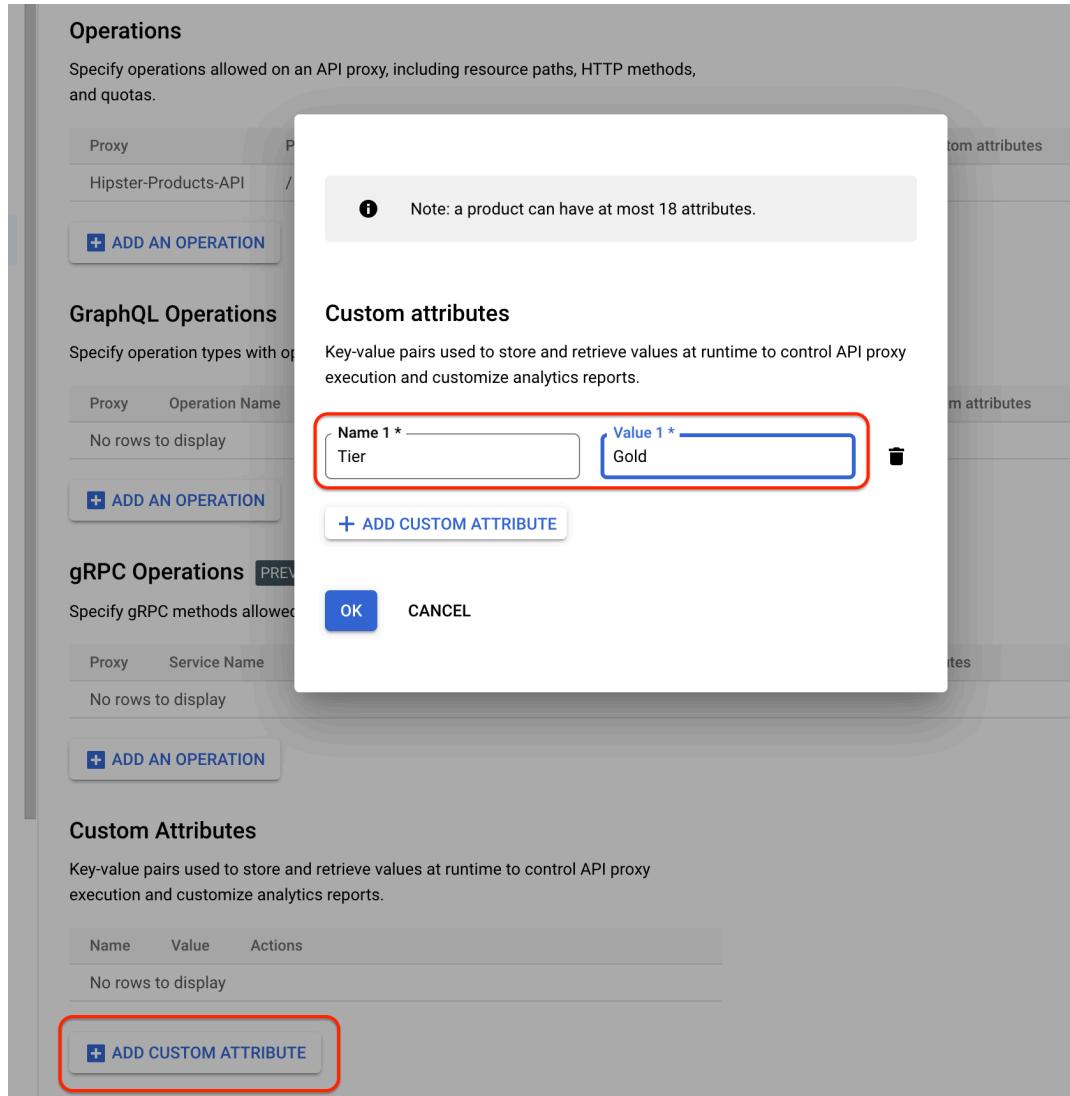
```

< Hipster-Products-API
OVERVIEW DEVELOP DEBUG
REVISION: 19 Updated a few seconds ago DEP
resources / jsc / Resource-1.js
Resource-1.js jsc
Used in
Policy: JS-
Resource-1.js
Press Option+F1 for Accessibility Options.
1 if(response.content)
2 {
3     var data = JSON.parse(response.content);
4     var tier = context.getVariable("verifyapikey.Verify-API-Key-1.apiproduct.Tier");
5
6     if(!tier || tier !== "Gold")
7     {
8         for(var i=0;i < data.products.length; i++)
9         {
10             delete data.products[i].priceUsd;
11             delete data.products[i].categories;
12         }
13     }
14     context.proxyResponse.headers['tier'] = tier;
15     context.proxyResponse.content = JSON.stringify(data);
16 }

```

8. [SAVE] をクリックし、[SAVE AS NEW REVISION] をクリックします。
9. [DEPLOY] をクリックします。[Deploy Hipster-Products-API] ペインが表示されたらすぐに、最新のリビジョンと適切な環境 (eval である必要があります) を選択します。[Deploy] をクリックします。
10. 次に、Gold の API 製品を編集する必要があります。[Distribution] -> [API Product] の下にあります。Hipster-Products-API-Product-Gold を見つけてクリックします。[Edit] をクリックし、[API Product] ページの最後までスクロール

し、[Custom Attributes] セクションで [ADD CUSTOM ATTRIBUTE] をクリックします。ポップアップが表示されたら、[Name 1] に [Tier] 、[Value 1] に [Gold] と入力します。[Custom attributes] ページで [OK] をクリックします。



上にスクロールして、[API Products] ページの [SAVE] をクリックします。

11. [オプション] Silver Productでも同じことができますが、この場合は違いはありません。これは、ティアがゴールドでない場合、上記の Javascript コードは **priceUsd** と **categories**を削除するだけであるためです。

## 3E - 階層化された製品のクオータと応答をテストする

次に、JavaScript と製品クオータをテストしてみましょう。今回は Apigee Debug Tool を使用します。

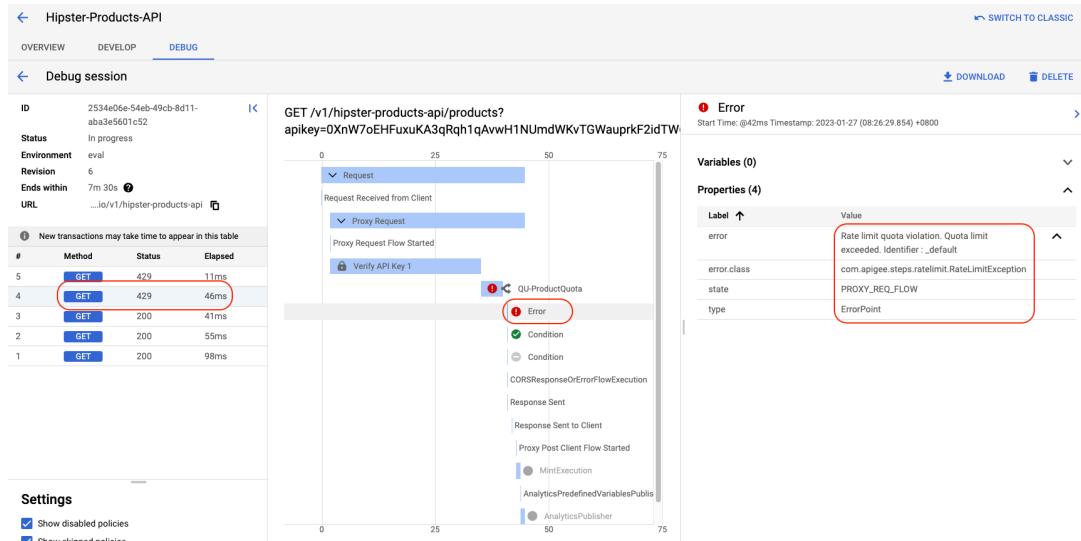
1. API プロキシの [Debug] タブに移動します。
2. [START DEBUG SESSION] をクリックします。
3. デプロイされた API リビジョンが [Environment] ドロップダウンで選択されていることを確認します。
4. [START] ボタンをクリックします。デバッグ セッションが開始されるまで待ちます。
5. POSTMAN や cURL コマンドなどのツールを使用して、次の URL にリクエストを送信します:

`https://{{API hostname}}/v1/hipster-products-api/products?apikey={{API Key}}`

上記 URL では:

- {{API hostname}} は、以前に [Admin] >> [Environments] >> [Groups] >> あなたの environment group からコピーした環境グループのホスト名です。
  - {{API Key}} は、シルバー層用に新しく作成したアプリ、つまり **Hipster Android App Basic** からコピーした API キーです。
6. **priceUsd** と **categories** のない Hipster 製品のリストが表示されるはずです。
  7. 繰り返し同じリクエストを複数回実行します。
- 5 回の呼び出しの後、5 回の呼び出しの無料割り当てを超えていていることがわかり、

割り当てポリシーに赤い感嘆符の記号が表示されます。



8. 次の API 製品レベル (Gold) をテストするには、[Premium] アプリ用に生成された API キーを使用します。apikey パラメータをPremium Appの認証情報と一致するように変更します。
9. Hipster Products の完全な属性 (priceUsdとcategoriesを含む) を取得しているはずです。この (Gold) レベルでは、1 分あたり最大 50 件まで呼び出することができます。

## Part 3 - まとめ

ラボのこの部分では、API 製品戦略に合わせて別の製品を作成し、階層型モデルを提供し、各プロダクトに異なるクオータを割り当てました。API プロキシでは制限を定義していませんが、クオータ量を定義するさまざまな API 製品で同じプロキシを利用できるようにしています。また、JavaScript ポリシーを使用して、層に基づいて応答を変更し、クライアントに応答を送り返しました。この手法を使用すると、単一の API プロキシを再利用するだけで、複数の API 製品で使用できるようになります。次の表は、API 製品 (それぞれの割り当て/属性が構成されている)、アプリ、および影響を受けるレスポンスをまとめたものです。

API Product	Quota in Product / Custom Attribute	Apps	Modified Response
Hipster-Products-API-Product-Silver	Quota: 5 per minute; Custom Attr: N/A	Hipster-Android-App-Basic	Remove priceUsd and categories
Hipster-Products-API-Product-Gold	Quota: 50 per minute; Custom Attr: Tier (Gold)	Hipster-Android-App-Premium	N/A

## 参考文献

### Apigee ドキュメント

- [Quota ポリシー](#)
- [JavaScript ポリシー](#)
- [JavaScript オブジェクトモデル](#)

## ディスカッションのための質問

1. アプリの API キーの有効期限や期間を強制するにはどうすればよいですか？
2. Javascript ポリシー（および Python または Java ポリシー）を使用すると、必要なロジックやコードを自由に記述できるという意味ですか？どのような影響がありますか？
3. Apigee には、[Assign Message ポリシー](#)と呼ばれる別のポリシーもあります。これで同様の結果を達成できるでしょうか？

# パート 4 - Apigee の統合開発者ポータルを使用したアプリデベロッパー エクスペリエンスの構築

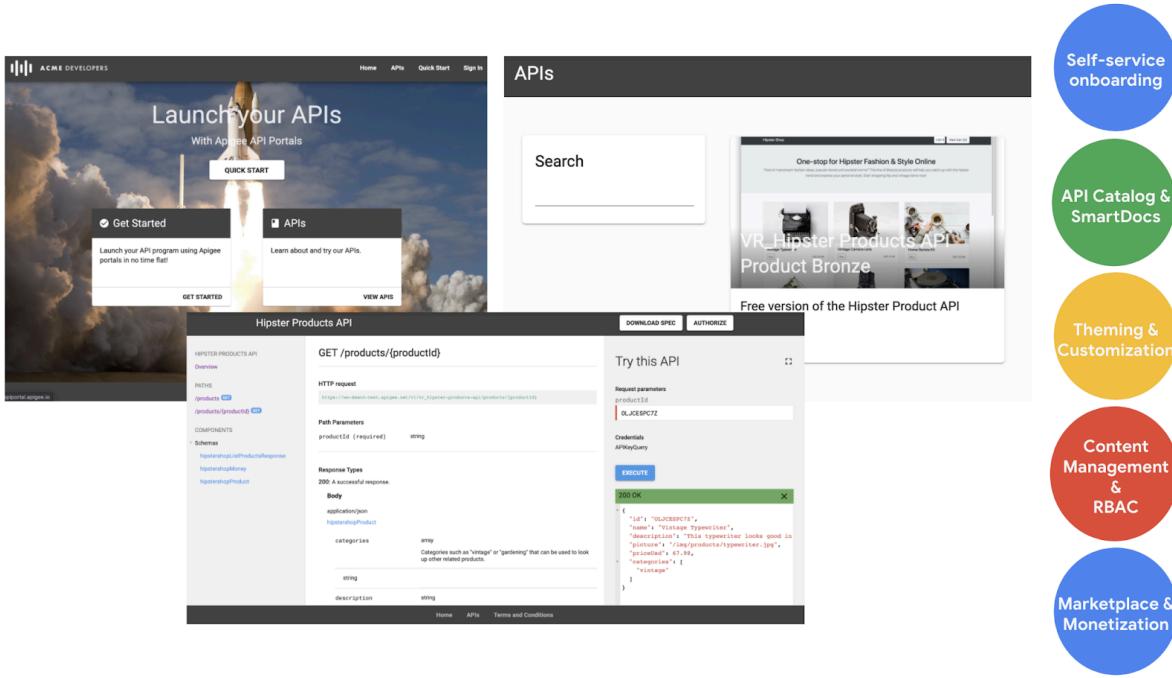
ペルソナ : API チーム

## ユースケース

開発者ポータル経由で API 製品を利用したいアプリ開発者に、簡単なセルフサービスのオンボーディング エクスペリエンスを提供および管理したいと考えています。アプリ開発者が API について学び、登録し、使用を開始できるようにするだけでなく、さまざまな API 製品への表示とアクセスを制御できるようにしたいと考えています。

## Apigee はどのように役立つでしょうか？

Apigee には、デベロッパー ポータル用の複数のオプションが用意されています。Apigee は、簡単なターンキーから完全にカスタマイズおよび拡張可能なものまで、いくつかのデベロッパー ポータル ソリューションをサポートしています。[統合開発者ポータル](#) オプションは、テーマ、ロゴ、ページ コンテンツなど、サイトの大部分のブランディングとカスタマイズをサポートしており、管理 UI から直接数秒で公開できます。また、Drupal マーケットで入手可能な何百もの Drupal モジュールを完全に制御して活用したい場合は、[Drupal ベースのポータル](#)も提供します。あるいは、Apigee API を利用して[完全にカスタマイズされた独自のポータル](#)を構築することもできます。このラボでは、統合開発者ポータルに焦点を当てます。



このラボでは、API 製品を公開する統合開発者ポータルを作成し、それを通じてアプリ開発者が次のことができます。

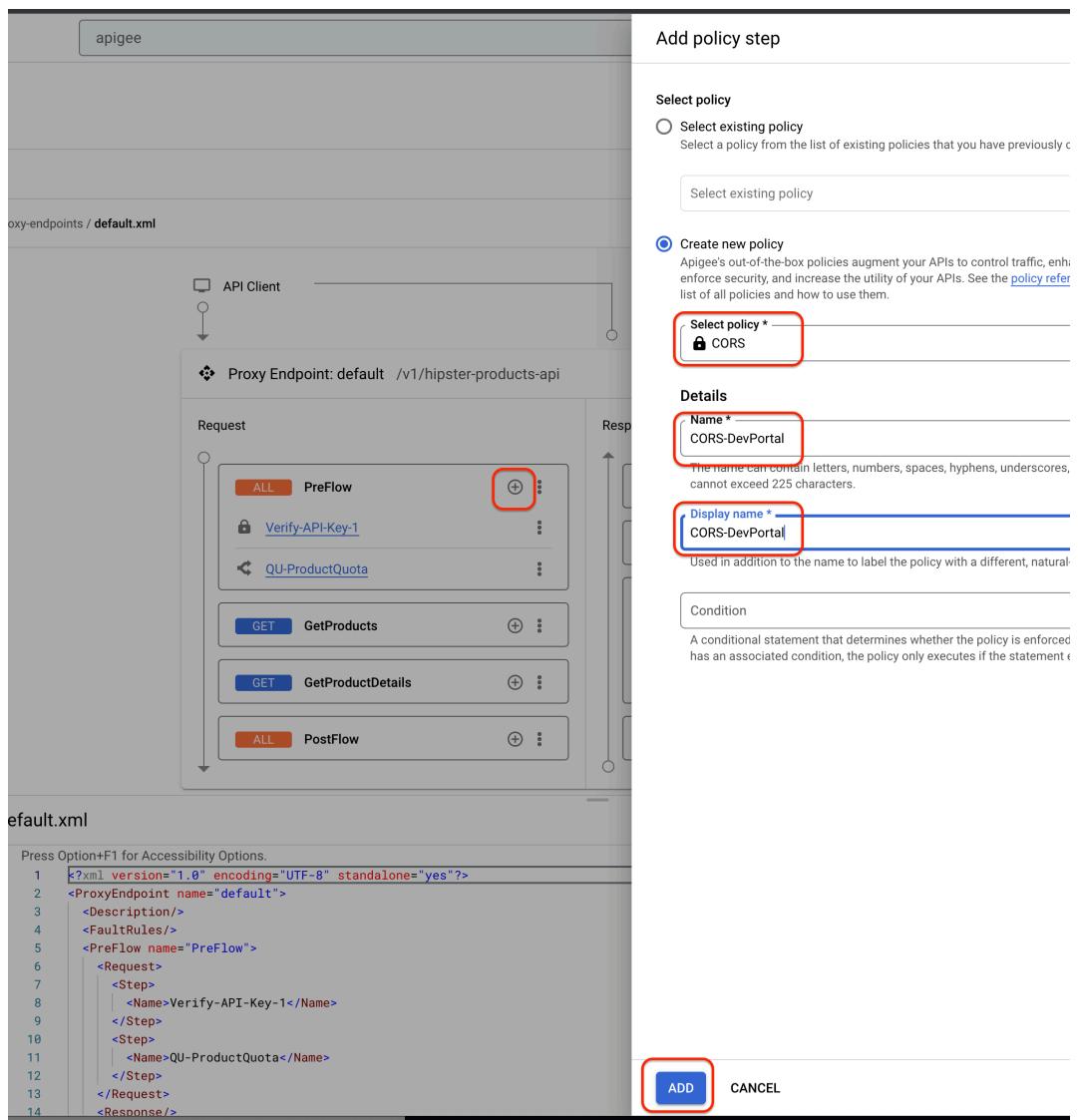
- OpenAPI仕様に基づいたインタラクティブなドキュメントを通じてAPIの使用法を学習する
- APIプロダクトを使用するアプリを登録する
- APIの使用に使用できるアプリクライアントの認証情報(APIキーとシークレット)を生成する

## パート 4A - CORS サポートのための API プロキシの更新

CORS (クロスオリジンリソース共有) は、Web ページで実行される JavaScript XMLHttpRequest (XHR) 呼び出しが非オリジンドメインのリソースと対話できるようにする標準メカニズムです。CORS は、すべてのブラウザによって強制される「[same-origin policy](#)」に対する一般的に実装されたソリューションです。たとえば、ブラウザで実行されている JavaScript コードから API プロキシへの XHR 呼び出しを行うと、呼び出しは失敗します。これは、ブラウザにページを提供するドメインが、API を提供するドメインと同じではありません。

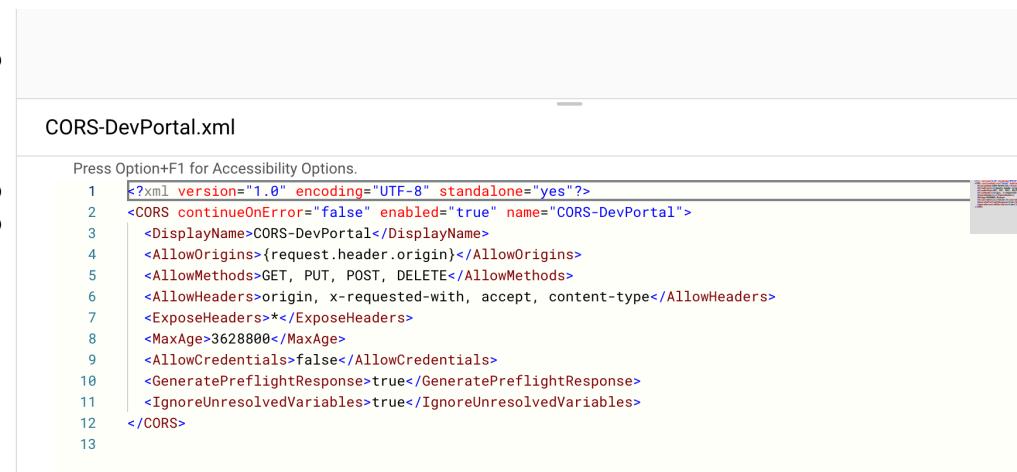
いためです。「{組織名}-{環境名}.apigee.net」。CORS は、クロスオリジンのリソース共有を提供したい場合にサーバーが「オプトイン」できるようにすることで、この問題の解決策を提供します。

1. Hipster-Products-API API プロキシに移動し、Proxy endpoints から default を選択していることを確認します。リクエストの PreFlow で (+) をクリックします。[Add policy] ステップが右側に表示されたら、すぐに [Select policy] ドロップダウンで CORS を見つけます。Name と Display name の両方に [CORS-DevPortal] と入力します。次に [ADD] をクリックします。



2. CORS-DevPortal ポリシーをクリックします。ペインの下部に CORS 構成 (XML 形式) が表示されていることがわかります。要件に合わせて値 (AllowOrigins、

AllowMethods など) を変更できます。今のところはそのままにしておきます。

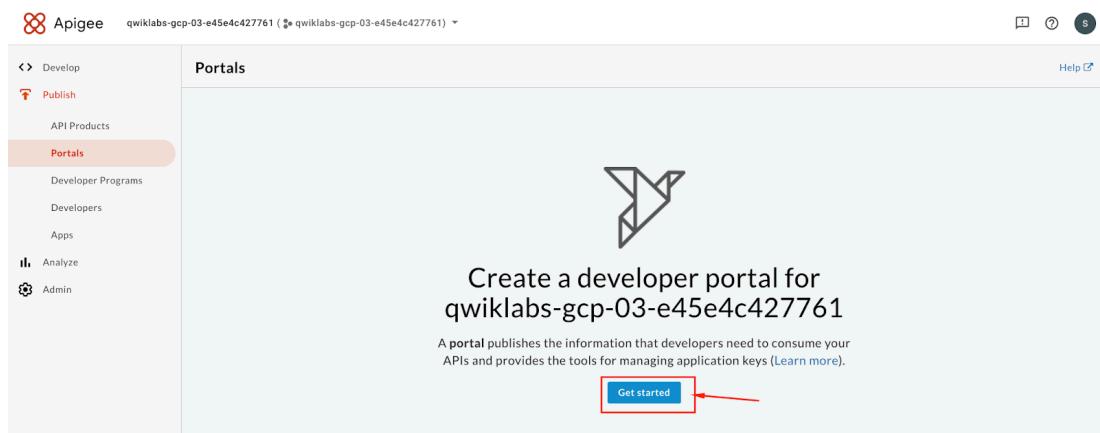


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CORS continueOnError="false" enabled="true" name="CORS-DevPortal">
  <DisplayName>CORS-DevPortal</DisplayName>
  <AllowOrigins>{request.header.origin}</AllowOrigins>
  <AllowMethods>GET, PUT, POST, DELETE</AllowMethods>
  <AllowHeaders>origin, x-requested-with, accept, content-type</AllowHeaders>
  <ExposeHeaders>*</ExposeHeaders>
  <MaxAge>3628800</MaxAge>
  <AllowCredentials>false</AllowCredentials>
  <GeneratePreflightResponse>true</GeneratePreflightResponse>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</CORS>
```

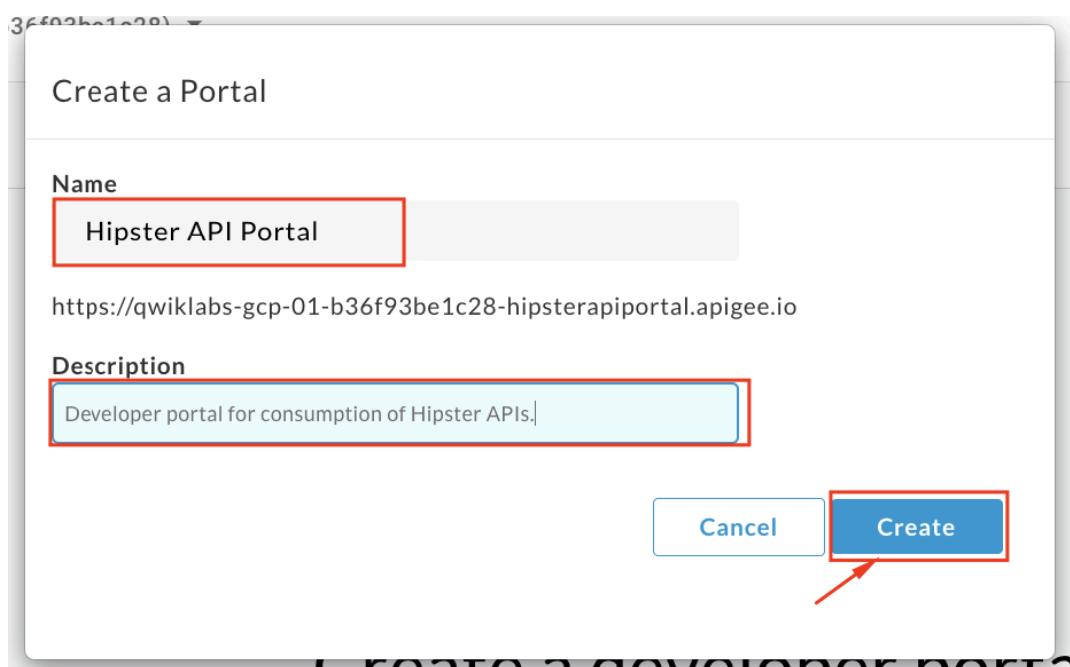
3. [SAVE] をクリックし、[Save as a new revision] として保存します。次に、最新のリビジョンを評価環境に [DEPLOY] します。

## パート 4B - 開発者ポータルの作成

1. このタスクでは、新しいブラウザ タブを開いて <https://apigee.google.com> と入力して、Apigee (クラシック) UI にアクセスする必要があります。あるいは、[Distribution] => [Portals] に移動し、[GO TO CLASSIC APIGEE UI] をクリックすることもできます。Apigee 組織が正しく選択されていることを確認してください。
2. [Publish] → [Portals] に移動し、[+ Portal] ボタンをクリックするか、[Get started] (組織内にまだポータルを作成していない場合) をクリックします。



3. ポータル作成ウィザードに詳細を入力します。
  - Name: **Hipster API Portal**
  - Description: **Developer portal for consumption of Hipster APIs.**
3. [Create] をクリックします。

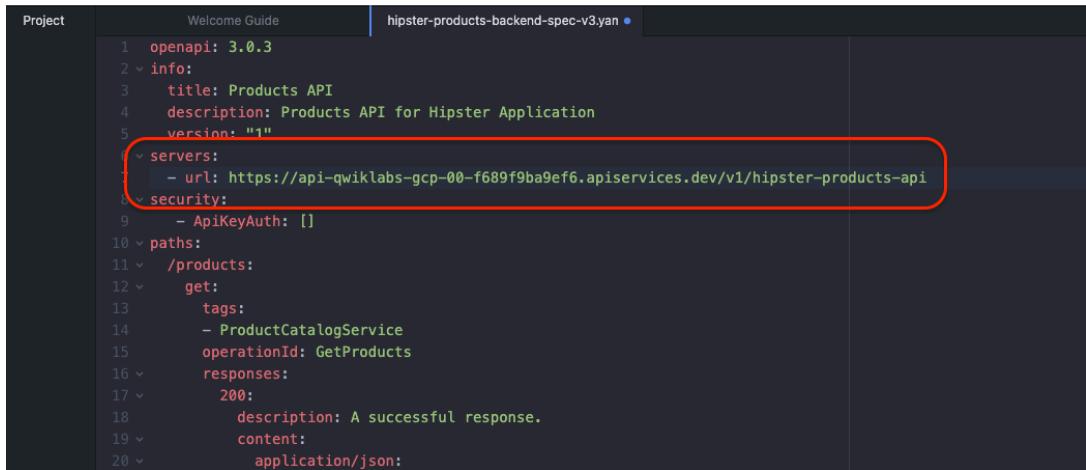


## 4C - OPEN API 仕様ファイルのホスト名を更新する

API ドキュメントを開発者ポータルに公開するので、パート 1A でダウンロードした Open API 仕様ファイル内のサーバー URL を更新する必要があります。

1. yaml ファイルを見つけます (ファイル名は `hipster-products-backend-spec-v3.yaml` である必要があります)。
2. 好みのテキストエディタでファイルを開きます
3. **servers - url**をアンコメントします。 [**servers - url**] の下に次の値を入力します:  
`https://{{API hostname}}/v1/hipster-products-api`

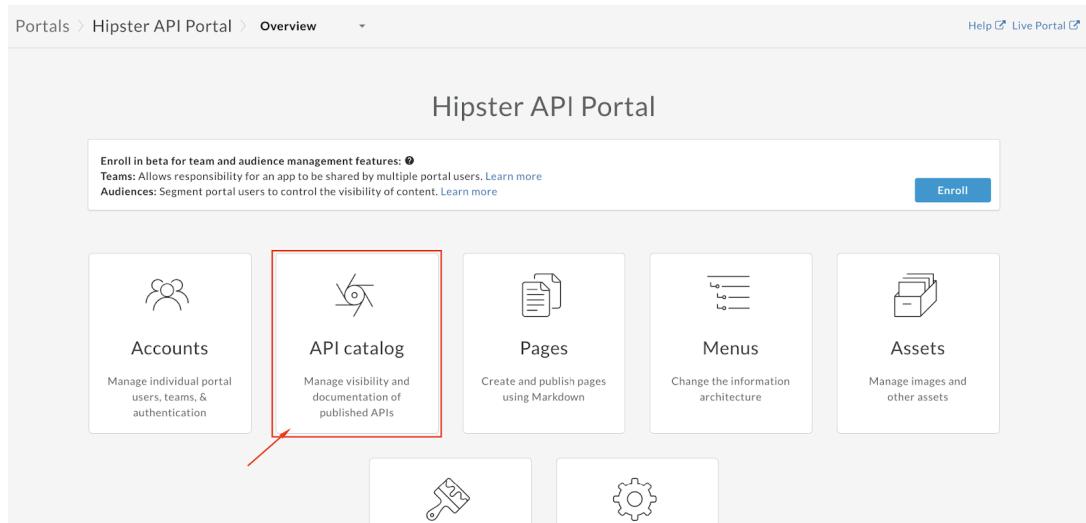
4. yaml のインデントに注意してください。以下に例を示します:



```
Project Welcome Guide hipster-products-backend-spec-v3.yaml •
1 openapi: 3.0.3
2 info:
3   title: Products API
4   description: Products API for Hipster Application
5   version: "1"
6   servers:
7     - url: https://api-qwiklabs-gcp-00-f689f9ba9ef6.apiservices.dev/v1/hipster-products-api
8   security:
9     - ApiKeyAuth: []
10  paths:
11    /products:
12      get:
13        tags:
14          - ProductCatalogService
15        operationId: GetProducts
16        responses:
17          200:
18            description: A successful response.
19            content:
20              application/json:
```

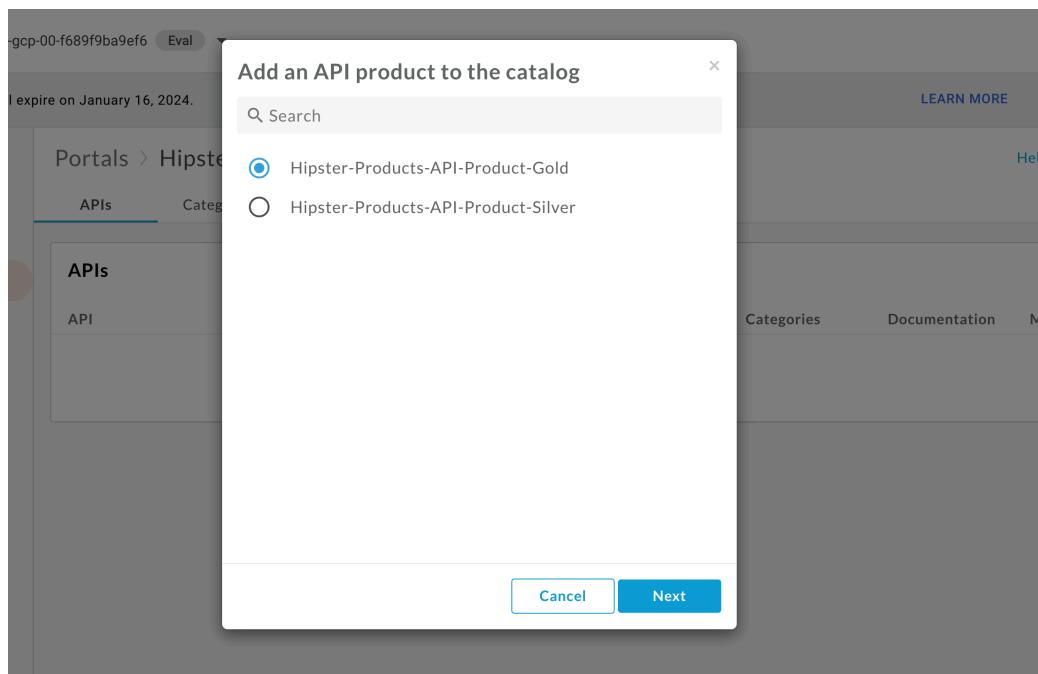
## 4D - ゴールド API 製品をポータルに公開する

1. ポータルオーバービューページで、[API catalog] ボタンをクリックします。

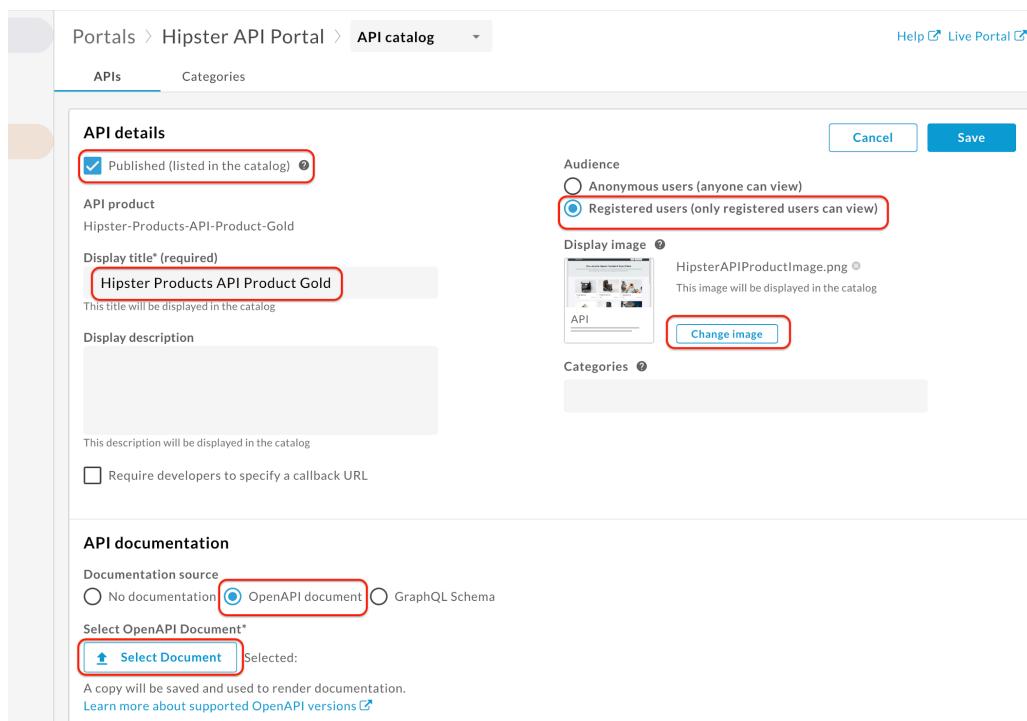


2. [+] ボタンをクリックし、ポータルに公開するゴールド API 製品を選択します。公開する [Hipster Products API Product Gold] を選択し、[Next] をクリックしま

す。

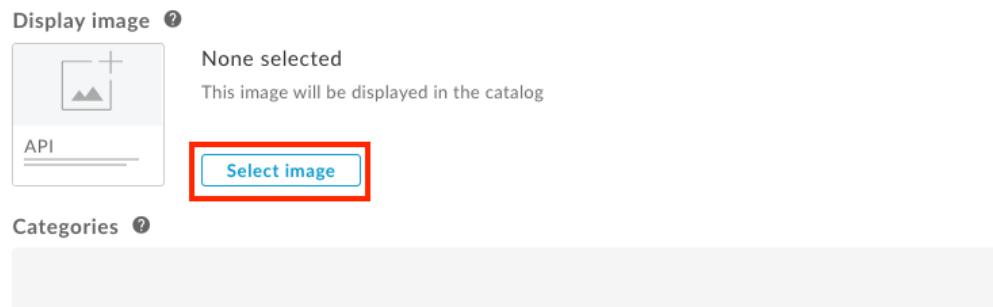


3. [Published] チェックボックスがオンになっていることを確認して、アプリの作成中に API カタログを通じて認定されたアプリ開発者にゴールド API プロダクトが表示されるようにします。



4. [Registered users] オプションを選択すると、開発者ポータルに登録されている開発者がポータルを通じてこの API を表示できるようになります。

5. [Select image] ボタンをクリックして、この API 製品に関連付けられたアイコン画像を更新します。



6. 次に、[Image URL] を選択し、次の URL を入力して画像をインポートします。

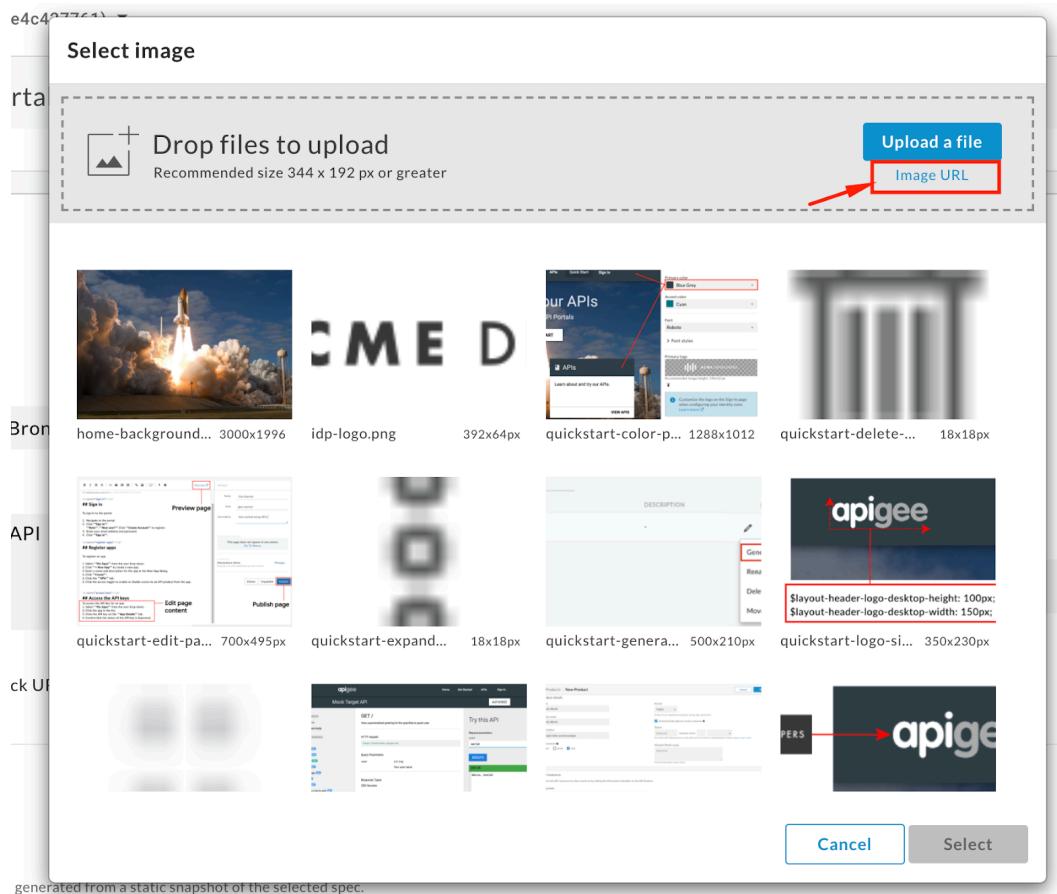
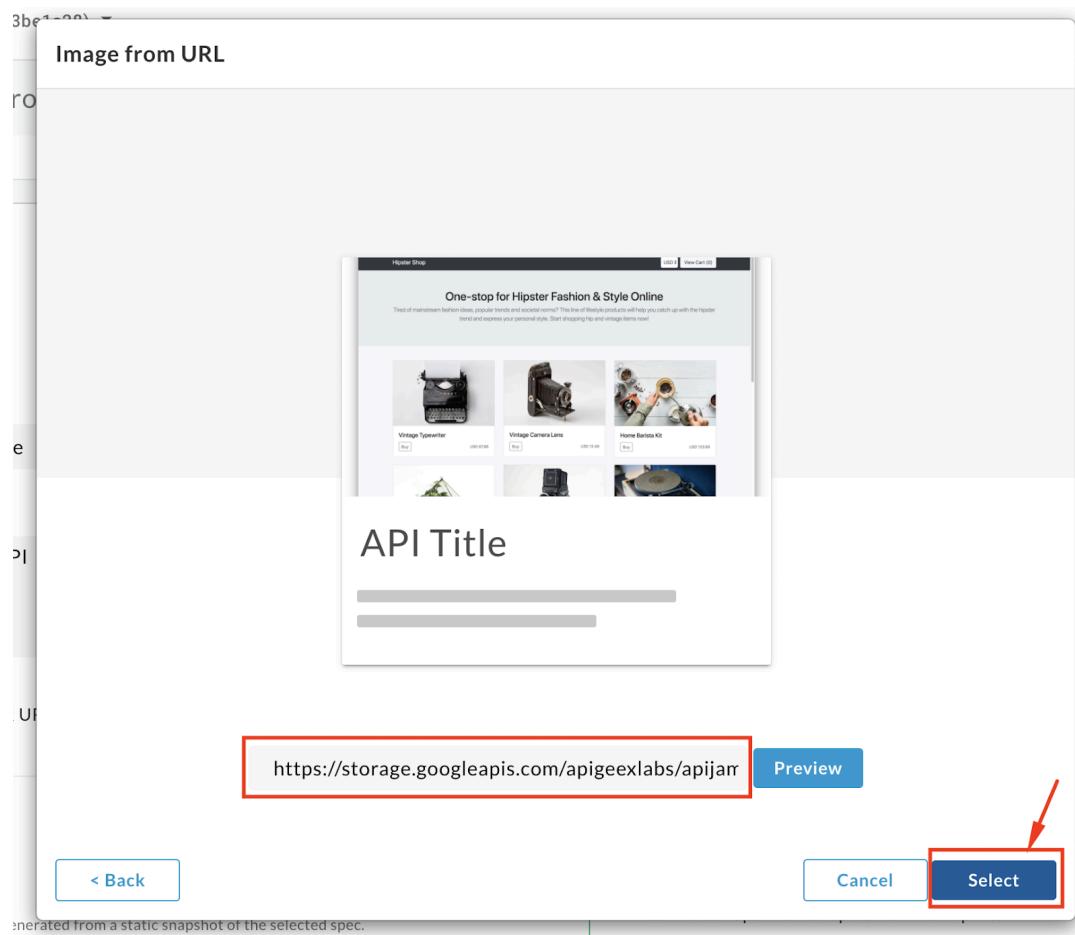


Image URL:

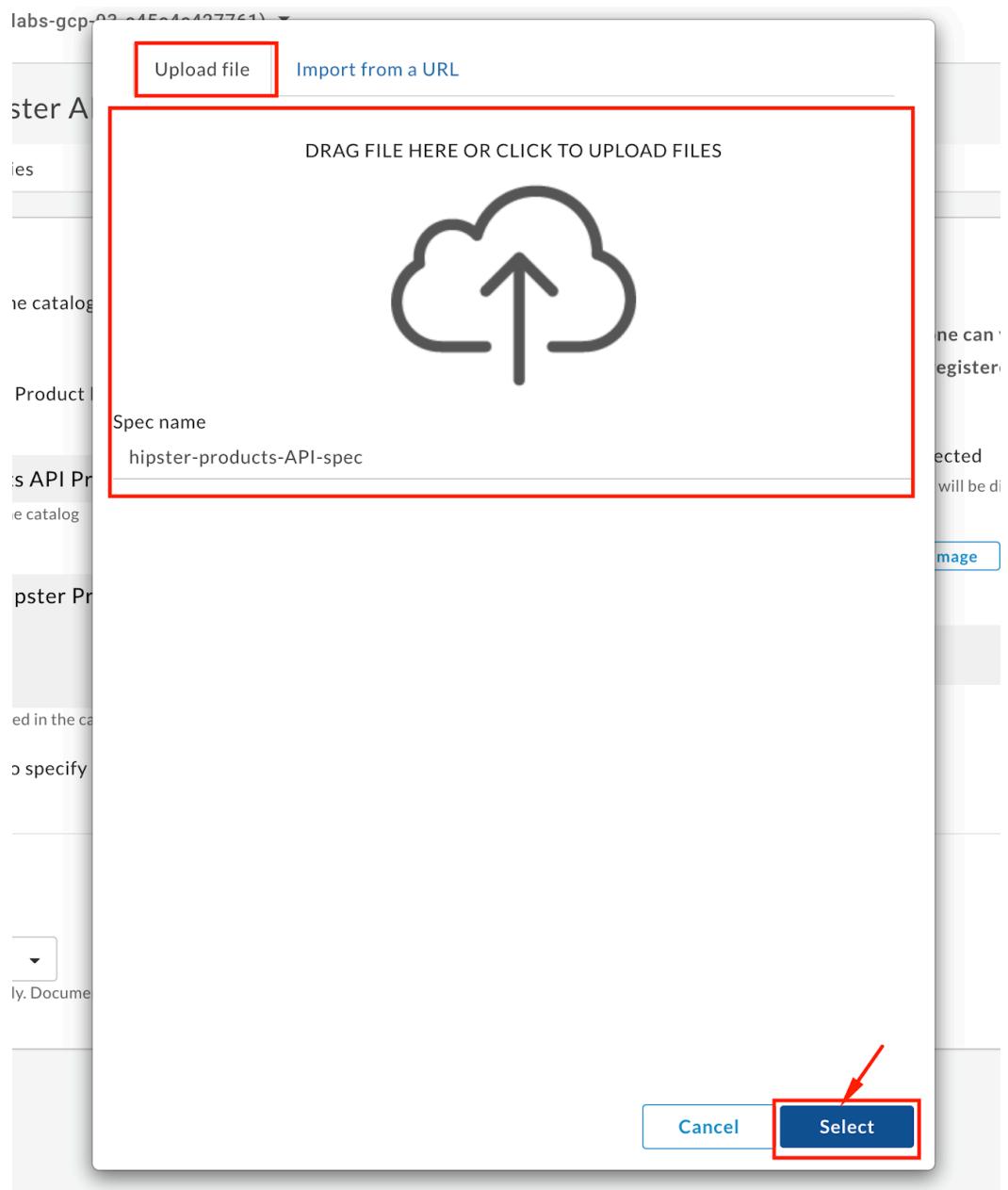
<https://storage.googleapis.com/apigeeexlabs/apijam-lab1/HipsterAPIProduct1>

mage.png



7. ページの下部に移動します。[API Documentation] で、[OpenAPI Document] を選択します。[Select Document] をクリックします。
8. 前の手順で作成および変更したローカルの [hipster-products-backend-spec-v3.yaml] ファイルをアップロードします。次

に、[Select] をクリックします。



9. 次に、[Save] をクリックして API 製品を保存し、API カタログに公開します。

API 製品が開発者ポータルに公開されました。

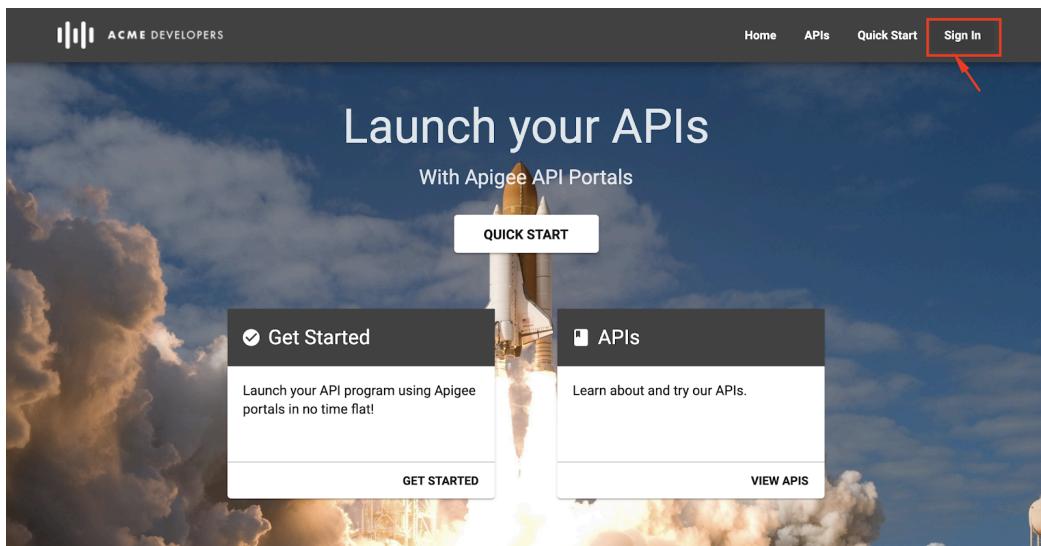
## 4E - アプリ開発者のサインアップ

ステップ 2C で、GCP Console (管理者の観点から) からアプリ開発者を作成しました。このステップでは、開発者ポータルを介したセルフサービス登録を許可します。管理者は、自己登録にモデレーションまたは承認のステップが必要かどうかを制御できます。

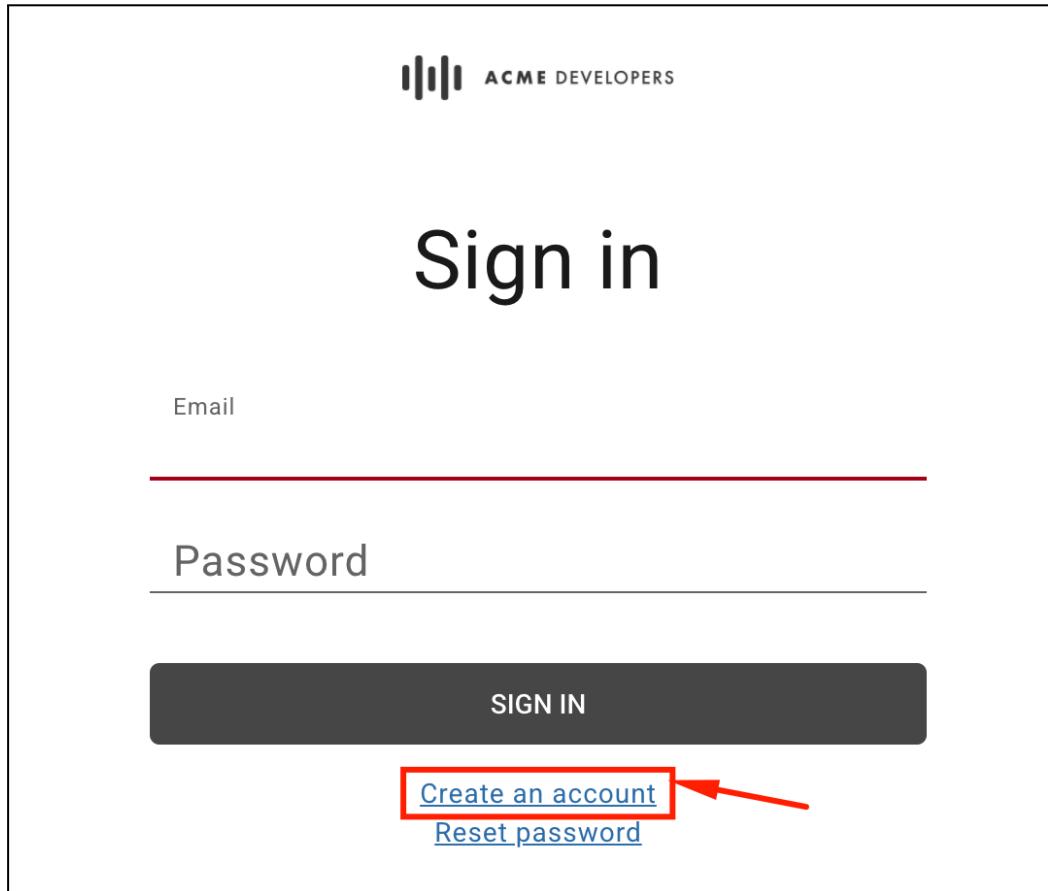
1. [Live Portal] リンクをクリックして、ブラウザのタブ/ウィンドウを起動し、新しい開発者ポータルを表示します。

A screenshot of the GCP API details page for 'VR\_Hipster-Products-API-Product-Bronze'. The URL is 'APIs > VR\_Hipster-Products-API-Product-Bronze'. At the top right, there is a 'Help' link and a red box highlighting the 'Live Portal' link. Below the header, there are tabs for 'APIs' (which is selected) and 'Categories BETA'. A section titled 'API details' is visible.

2. 開発者ポータルで、[Sign In] というラベルの付いたメインメニュー オプションをクリックします。



3. これにより、アプリ開発者のログインページが表示されます。ここで、[Create an Account] をクリックします。



4. 次の詳細を入力し、[Create Account] をクリックします。名: {{あなたの名}} 姓: {{あなたの姓}} 電子メール: {{あなたの電子メール アドレス}} パスワード: {{パスワードを入力}} [I agree to terms.] ボックスにチェックを入れます。



# Create your account

First Name

Last Name

Email

Password

••••••••



I agree to the terms.

Create Account

Have an account? [Sign in.](#)

いくつかのcaptcha画像を確認する必要がある場合があります。最後に [Verify] をクリックします。

- アカウントが作成時されると、アプリ開発者はアカウント確認リンクが記載された電子メール通知を受け取ります。



# Check your inbox

We sent a confirmation email to you.

Nothing in your inbox? [Resend link](#)  
(Be sure to check your spam folder.)

このラボではアプリ開発者として自分の電子メール アドレスを指定したため、この通知を受け取っているはずです。リンクをクリックするか、コピーしてブラウザに貼り付けてアカウントを確認します。

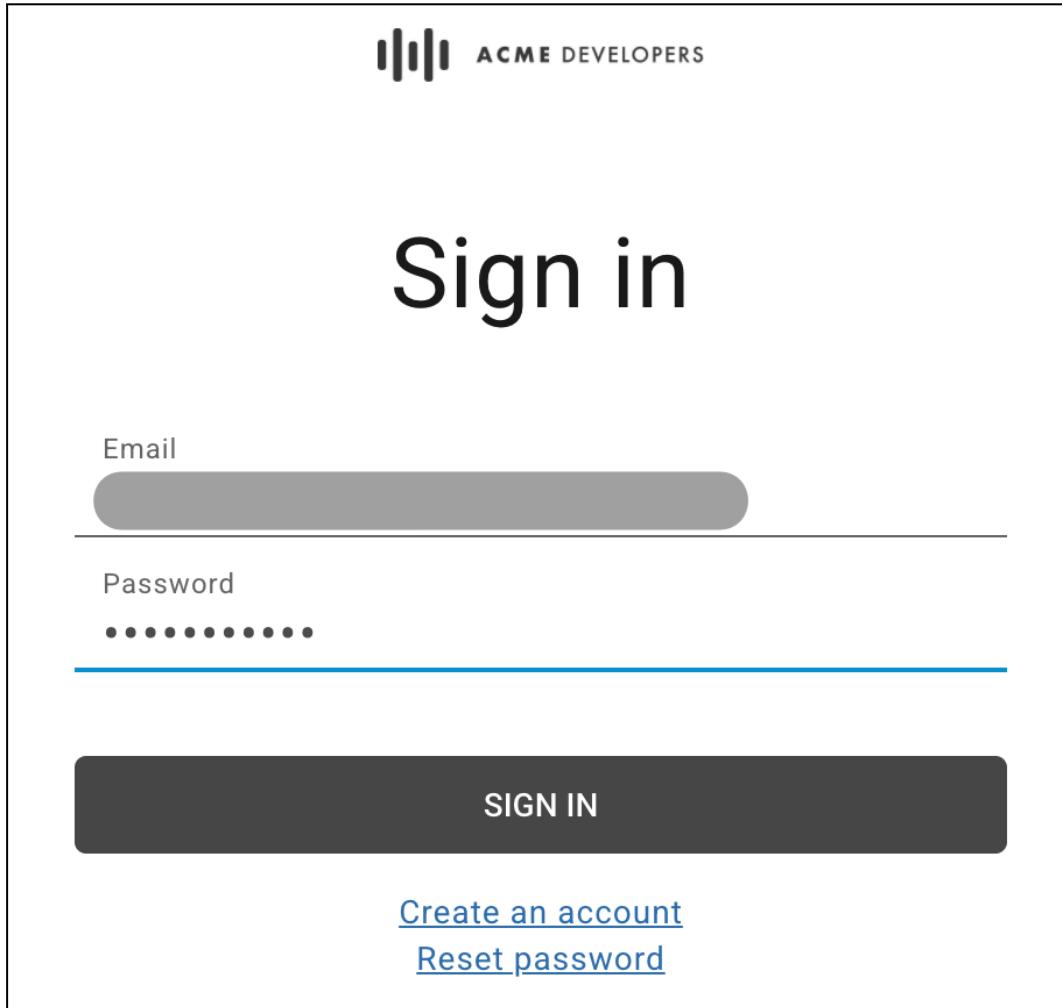
Verify your new account ➔ Inbox X 🖨️ 📎

**qwiklabs-gcp-01-b36f93be1c28-hipsterapiportal** <no-reply@google.com> • to me ▾ 12:58 PM (0 minutes ago) ☆ ↶ ↷ ⋮

Welcome Vidya, Thank you for signing up for an account. To verify your identity, please visit this link within 10 minutes: [https://accounts.google.com/signin/identifier?hl=en&passive=1209600&service=account&source=gmail&continue=https%3A%2F%2Fqwiklabs-gcp-01-b36f93be1c28-hipsterapiportal.appspot.com%2Faccounts%2Fconfirm&reauth=false&utm\\_source=qwiklabs-gcp-01-b36f93be1c28-hipsterapiportal&utm\\_medium=email&utm\\_campaign=welcome&utm\\_term=.0&utm\\_content=link](https://accounts.google.com/signin/identifier?hl=en&passive=1209600&service=account&source=gmail&continue=https%3A%2F%2Fqwiklabs-gcp-01-b36f93be1c28-hipsterapiportal.appspot.com%2Faccounts%2Fconfirm&reauth=false&utm_source=qwiklabs-gcp-01-b36f93be1c28-hipsterapiportal&utm_medium=email&utm_campaign=welcome&utm_term=.0&utm_content=link)

↶ Reply ↷ Forward

- このアカウントが検証されると、資格情報を使用してポータルにアプリ開発者としてサインインできるようになります。



## 4F - API ドキュメントの表示、サンプルコードの生成、および API のテスト

- 前の手順で作成したアカウント資格情報を使用して、アプリ開発者としてログインします。開発者ポータルの [API] メニュー リンクをクリックします。API カタログ ページに移動します。ここには、登録されているすべての開発者に表示されるように以

前に公開された API 製品が表示されます。

The screenshot shows the ACME Developers API catalog. At the top, there are navigation links: Home, APIs (which is highlighted with a red box and circled with a red arrow labeled 1), Quick Start, and a user account link. Below the navigation is a search bar with a placeholder 'Filter by title & description' and a magnifying glass icon. The main content area displays a card for the 'Hipster Products API Product'. The card features a thumbnail image of a mobile application interface, the product name 'Hipster Products API Product', its tier 'Bronze', and a brief description 'Free version of the Hipster Product API'. A red box highlights the product card, and a red arrow labeled 2 points from the left side of the card towards the right panel.

2. カタログ内の Gold 製品の API Product アイコンをクリックして、そのドキュメントを表示します。これにより、公開時に API 製品に関連付けられた OpenAPI 仕様から生成された対話型のドキュメントページが表示されます。

The screenshot shows the detailed documentation for the Hipster Products API. The top navigation bar includes Home, APIs, Quick Start, and a user account link. The main title is 'Hipster Products API'. On the left, there's a sidebar with 'HIPSTER PRODUCTS API' and sections for 'Overview' (highlighted in purple), 'PATHS', and 'RESOURCES'. Under 'PATHS', there are two entries: '/products' (with a 'GET' method) and '/products/{productId}' (with a 'GET' method). Under 'RESOURCES', there are three entries: 'hipstershopListProductsResponse', 'hipstershopMoney', and 'hipstershopProduct'. The right panel contains the 'Hipster Products API' documentation, which includes a brief description 'Products API for Hipster Application', a 'Resources' section, and a table for the 'hipstershopListProductsResponse' resource. The table has columns for 'Method', 'Endpoint', and 'Description', with one row for 'GET /products'. Below this, there are sections for 'hipstershopMoney' and 'hipstershopProduct'.

3. ドキュメントの左側のパネルから API リソース パスの 1 つを選択し、[Execute] をクリックして API をテストします。その後、右側のパネルに Unknown Error や 401 Unauthorized responseなどのエラーが表示される場合があります。

The screenshot shows the Apigee API developer portal interface. On the left, there's a sidebar with 'PATHS' and 'RESOURCES' sections. The main area displays the 'GET /products' endpoint details. A red box highlights the 'EXECUTE' button. To the right, a 'Try this API' section is shown with a 'Request parameters' field (empty), 'Credentials' (APIKeyQuery), and a 'Response Types' section indicating a successful 200 response. A red box highlights the 'EXECUTE' button again. A modal window titled '401 Unauthorized' is open, showing the JSON response: 
 

```
{
      "fault": {
        "faultstring": "Failed to resolve API Key v2",
        "detail": {
          "errorcode": "steps.oauth.v2.FailedToResolve"
        }
      }
    }
```

これは、まだアプリを登録していないため、承認された API 呼び出しに使用する API キーがないためです。このエラーは次のパートで修正します。

4. [Try this AP] の横にある全画面表示アイコンをクリックします。ペインが展開されると、Apigee 開発者ポータル が cURL、HTTP、Python、Node.JS、JavaScript、PHP、Java などのさまざまなプラットフォームでサンプル スクリプト/コードを自動的に生成することに注目してください。

The screenshot shows the expanded 'Try this API' section. A red box highlights the tabs for different code samples: 'HTTP', 'PYTHON', 'NODE.JS', and 'JAVA'. Below the tabs, a code editor displays sample Node.js code for making a GET request to the products API. The code uses the 'request' module and includes comments explaining the purpose of the code.

```
// Demo code sample. Not intended for production use.

// See instructions for installing Request module for NodeJS
// https://www.npmjs.com/package/request

const request = require('request');

function execute() {
  const options = {
    "url": "https://api-qwiklabs-gcp-03-6f28ba31d705.apiservices.dev/v1/hipster-products",
    "method": "GET",
    "headers": {
      "Accept": "application/json"
    }
  };
  request(options, function (err, res, body) {
    if (err) {
      console.error(err);
    } else {
      console.log(body);
    }
  });
}

execute();
```

の機能により、アプリ開発者はサンプル コードをエディタ/IDE に簡単にコピー アンド ペーストして、API を簡単にテストできるようになります。

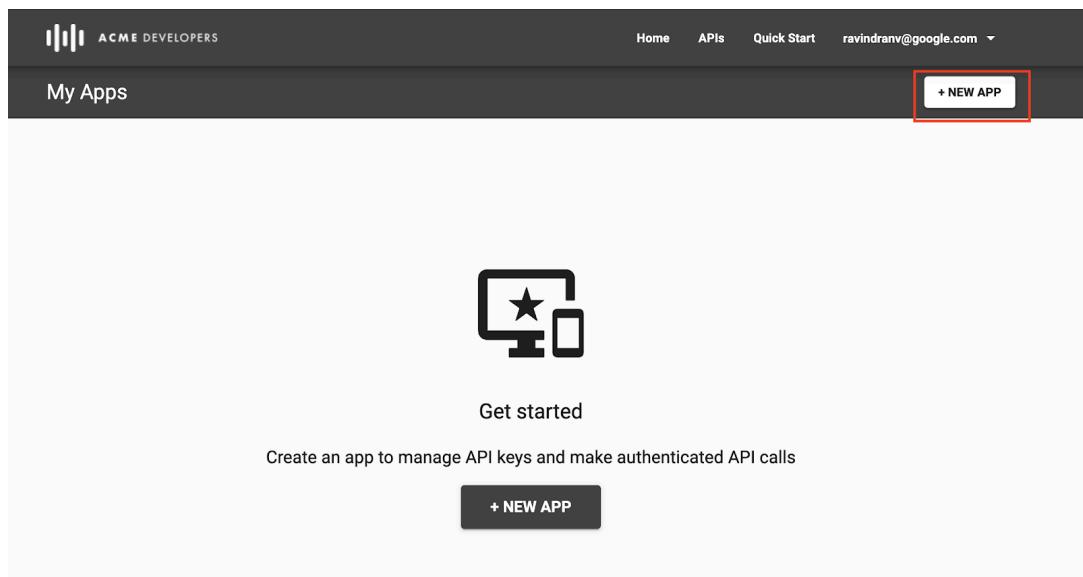
## 4G - セルフサービスアプリの登録

アプリ開発者の作成プロセスと同様に、アプリ開発者はセルフサービス方式で開発者ポータルにアプリを登録することもできます。

- 右上隅にある開発者アカウントのドロップダウンメニューに移動し、[Apps] リンクを選択します。



- [New App] ボタンをクリックします (以下に示すように、ページ自体または上部パネル上)。



- 次のアプリの詳細を入力し、[Create] をクリックします:
  - App Name: Hipster-Test-App
  - Description: Test app to try out Hipster Products API using the Gold API Product
  - サブスクリプション可能なGold API 製品を選択します。[Enable] をクリックし、[Save] をクリックします。

4. 新しく作成したアプリに対して API キーとシークレットのペアが生成されたことがわかります。これで、この API キーを使用して API をテストできるようになります。

The screenshot shows the 'Hipster-Test-App' configuration page. Under the 'API Keys' section, there is a table with one row. The row contains a redacted API key, a 'Copy API key' button, a 'Secret' column (with a 'Show secret' link), a 'Status' column (Active), a 'Created' column (2021-02-16), an 'Expires' column (never), and an 'Actions' column (Revoke). Below the table is an 'ADD KEY' button.

5. API カタログに移動し、Gold API 製品ドキュメントを選択し、GET /products リソースを再度テストします。ただし今回は、まず [Authorize] ボタンをクリックし、新しく作成したアプリの資格情報を選択して、API 呼び出しの承認情報 (API キー) を設定します。

The screenshot shows the 'ACME DEVELOPERS' API catalog. The 'APIS' tab is selected. A red box highlights the 'Hipster Products API Product' card, which is labeled 'Bronze' and described as the 'Free version of the Hipster Product API'. A red arrow points from the number '②' to this card. Another red box highlights the 'APIS' tab in the top navigation bar, with a red arrow pointing from the number '①' to it.

Hipster Products API

DOWNLOAD SPEC AUTHORIZE

HIPSTER PRODUCTS API

Overview

PATHS

/products (GET) /products/{productId} (GET)

COMPONENTS

Schemas

- hipstershopListProductsResponse
- hipstershopMoney
- hipstershopProduct

GET /products

HTTP request

<https://api-qwiklabel-gcp-01-b36f93be1c28.apigee-apisjan.dev/v1/hipster-products-api/products>

Response Types

200: A successful response.

Body

application/json

hipstershopListProductsResponse

products array

hipstershopProduct

id	string
name	string
description	string
picture	string

price float

hipsterShopMoney

Try this API

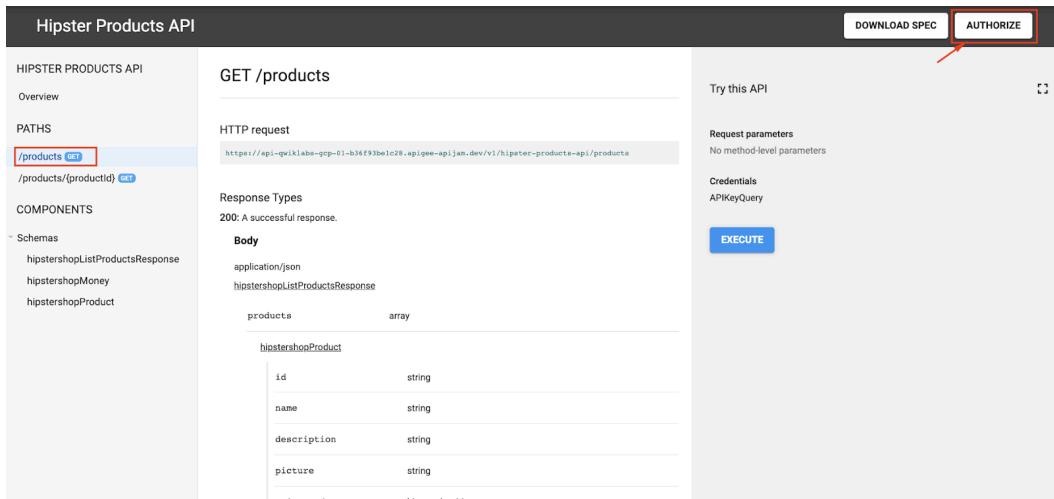
Request parameters

No method-level parameters

Credentials

APIKeyQuery

EXECUTE



## Authorization

Hipster Products API requires authorization. Enter your credentials to make calls to this API.

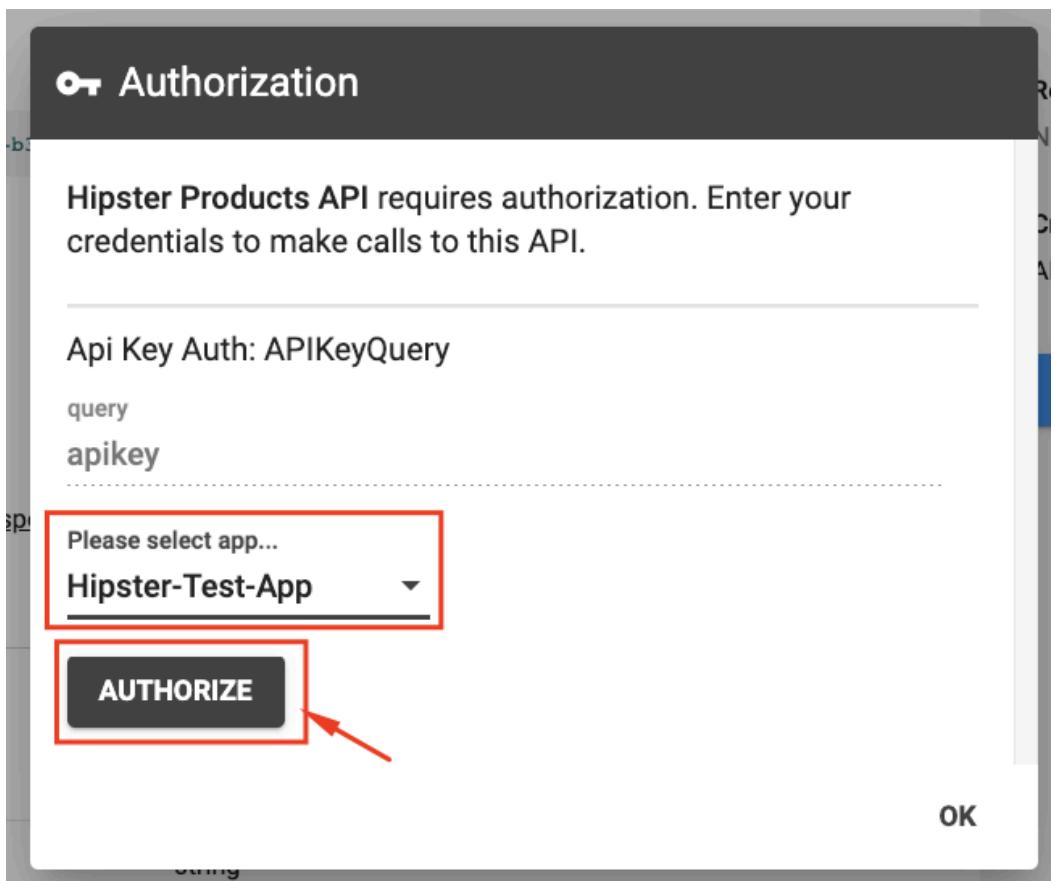
Api Key Auth: APIKeyQuery

query  
apikey

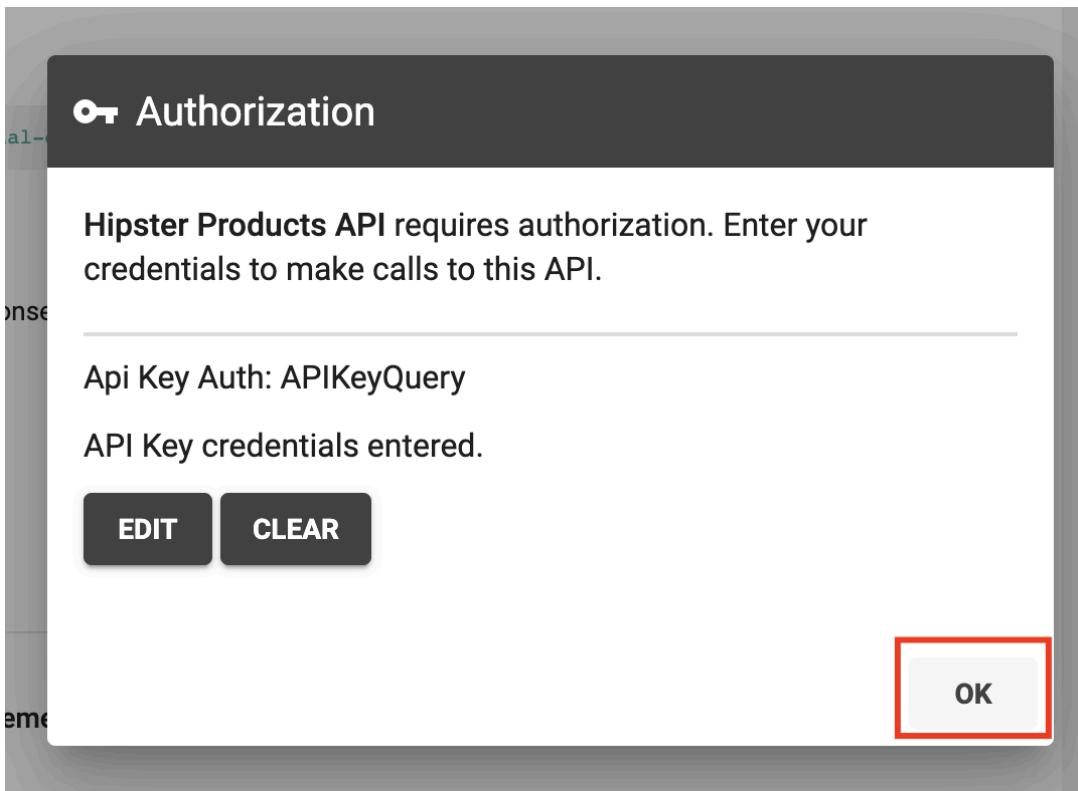
Please select app...  
Hipster-Test-App

AUTHORIZE

OK



認可情報を設定したら [OK] をクリックします。



6. 次に、[Execute] をクリックします。有効な 200 OK API Responseが受信されたことがわかります。

The screenshot shows a detailed API documentation page for the 'Hipster Products API'. On the left, there's a sidebar with 'HIPSTER PRODUCTS API' and sections for 'Overview', 'PATHS' (with '/products'), and 'RESOURCES' (including 'hipstershopListProductsResponse', 'hipstershopMoney', and 'hipstershopProduct'). The main content area shows the 'GET /products' endpoint. It includes an 'HTTP request' section with the URL 'https://ravindrantrial-eval-test.apigee.net/v1/vr\_hipster-products-api/products', 'Response Types' (a 200 status code), and a 'Body' section showing a JSON object with 'products' as an array. To the right, there's a 'Try this API' section with 'Request parameters' (none), 'Credentials' (APIKeyQuery), and a 'EXECUTE' button. A red box highlights the 'EXECUTE' button. A modal window titled '200 OK' is open, displaying a JSON response object:

```
{ "products": [ { "id": "OLJCESPC7Z", "name": "Vintage Typewriter", "description": "This typewriter looks good.", "picture": "/static/img/products/typewriter.jpg", "priceUsd": { "currencyCode": "USD", "units": "67", "nanos": 990000000 }, "categories": [ "vintage" ] } ] }
```

## Part 4 - まとめ

次の方法を学習しました:

- Apigee 統合開発者ポータルのデプロイ
- OpenAPI 仕様を使用して API 製品を公開する
- 開発者ポータル UI を使用して、API カタログ、ドキュメント、サンプル コードを参照する
- アプリを登録して、開発者として API 製品を利用する

## 参考文献

### Apigee ドキュメント

- [統合ポータルの構築](#)
- [チュートリアル：最初のポータル作成](#)
- [デベロッパー ポータル ソリューション](#)

## ディスカッションのための質問

1. 電子メールアドレスを表示するお問い合わせフォームなどの単純なページを追加し、それを開発者ポータルのナビゲーション メニューに追加するにはどうすればよいですか？
2. 開発者ポータルでアプリの別の認証情報セットを生成するにはどうすればよいですか？
3. 開発者ポータルにディスカッション フォーラム機能を追加したいと考えています。Apigee Integrated Portal でそのような機能を使用できますか？それ以外の場合、どのような選択肢がありますか？

## ボーナスチャレンジ [オプション]

以下を追加して、会社のテーマや外観に合わせて開発者を再設計します:

- ロゴ
- プライマリ&アクセントカラーを含む基本的なスタイル

- 背景イメージ

# パート 5 - Apigee Analytics を使用した API プログラム成功の測定

ペルソナ : API プロダクト Team

## ユースケース

最初の API 製品を公開した後に何が起こるか

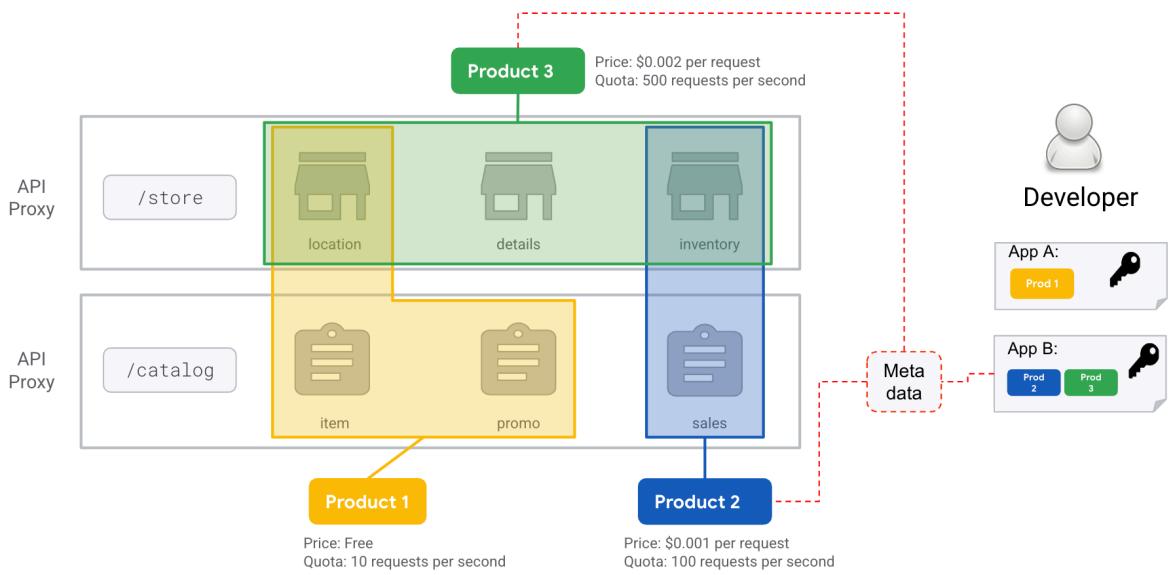
いくつかの API 製品を稼働させたら、プログラムがどのように実行され、API プログラムがどの程度成功したかを確認する必要があります。

- 時間の経過とともにトラフィックの傾向はどうなっていますか？
- トラフィックの流れが最も速くなるのはいつですか？
- 最もユーザーが多いのはどこですか？
- 最も活発な開発者は誰ですか？

インサイトはキャパシティプランニングと API プログラムに役立ちます  
使用状況の傾向データを使用すると、API 呼び出しの長期的な傾向を確認し、容量計画を行う必要がある来年のトラフィックがどうなるかを予測できます。

Apigee は、すぐに使える分析によるデータの視覚化を通じて、API の傾向とパターンを見つけるのに役立ちます。

トラフィックを API 製品に結び付ける



アプリケーションを API 製品に登録する場合、API キーまたは OAuth トークンを検証するときにメタデータが読み込まれるため、開発者、アプリケーション、製品を識別できます。このデータは、プログラムについての洞察を得るために非常に重要です。

**注:** \* 前のパートの手順に従った場合は、以前に作成したプロキシのトラフィックをすでに確認し、カスタム分析用に以前の API プロキシを使用できるはずです。\* 前のパートの手順に従っていない場合は、内容の確認自体はできますがダッシュボードに表示されるデータは少なくなります。

## パート 5A - 事前構築された Apigee 分析ダッシュボードを使用して API の使用状況を理解する

Apigee には、事前に構築されたレポートが多数付属しています。これらのラボの目的と時間の都合上、入力済みのデモ環境のスクリーンショットを使用してさまざまなダッシュボードを見ていきます。オプションのロードジェネレーターを自由に使用して、独自の環境で進めしてください。

必要に応じて、多くの値が入力されたメトリクス ダッシュボードの概要を説明するこのビデオを見て、API ダッシュボードを確認することができます：

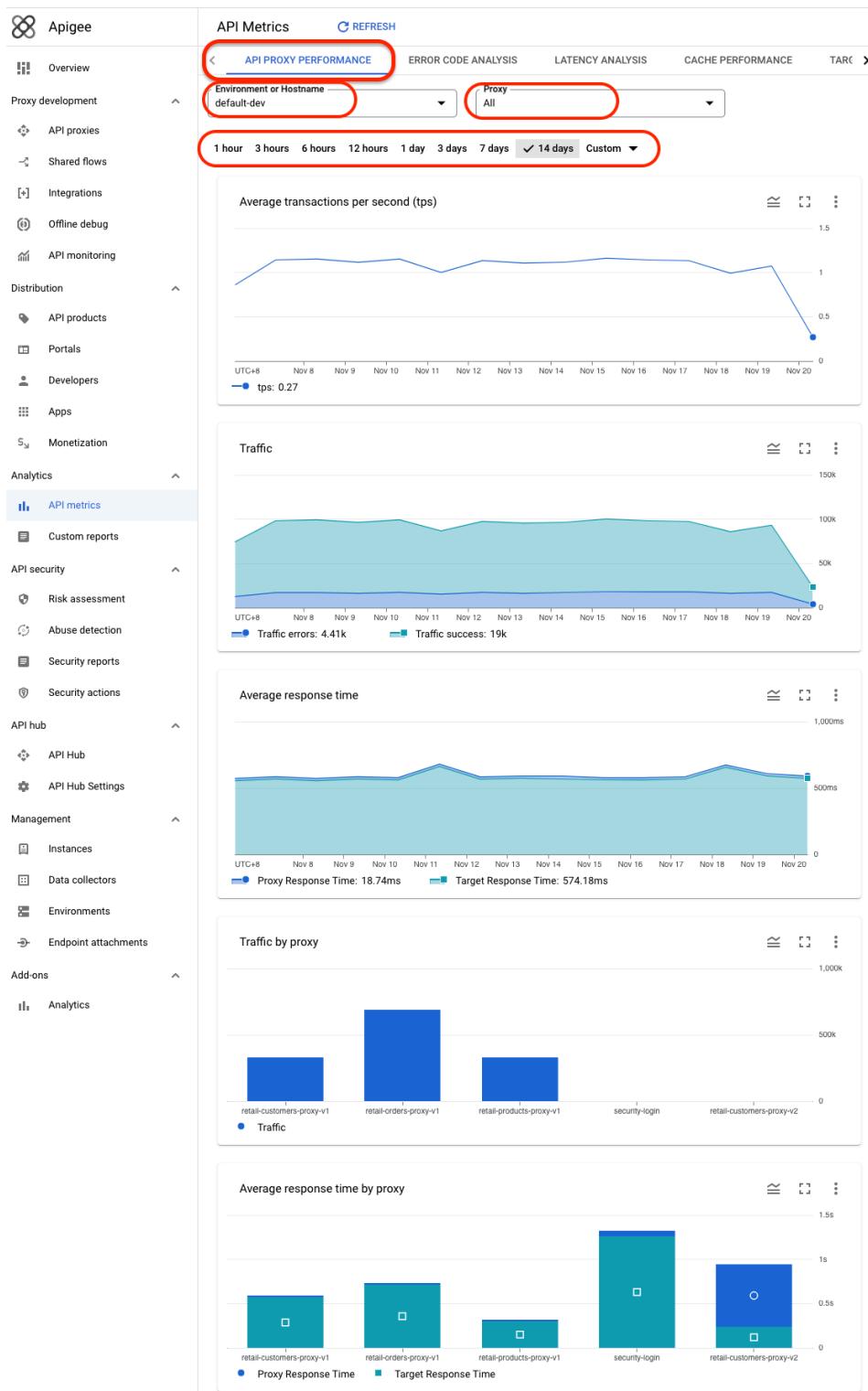
<https://www.youtube.com/watch?v=A2KQ3GJvb98&t=88s> 新しいコンソール/UI では若干異なって表示される可能性があることに注意してください。

レポートにアクセスするには、サイドバーの [Analytics] の下にある [API metrics] メニューを開きます。

The screenshot shows the Apigee API Metrics interface. On the left, there's a sidebar with sections for API products, Portals, Developers, Apps, Monetization, Analytics (with 'API metrics' highlighted by a red box), and API security. The main area is titled 'API Metrics' and has a 'REFRESH' button. It features two tabs: 'API PROXY PERFORMANCE' (which is selected) and 'ERROR CODE ANALYSIS'. Below these tabs are dropdown menus for 'Environment or Hostname' (set to 'default-dev') and 'Proxy' (set to 'All'). A time range selector shows '1 hour', '3 hours', '6 hours', '12 hours', '1 day' (which is checked), '3 days', and '7 days'. The main content area displays a chart titled 'Average transactions per second (tps)'.

## 1. API プロキシのパフォーマンス

プロキシ パフォーマンス ダッシュボードは、API プロキシトラフィックパターンと処理時間を確認するのに役立ちます。API が生成するトラフィックの量と、API 呼び出しが Apigee で受信されてからクライアント アプリに返されるまでの処理にかかる時間を簡単に視覚化できます。このダッシュボードでは、1 秒あたりの平均トランザクション (tps)、トラフィック、平均応答時間、プロキシごとのトラフィック、およびプロキシごとの平均応答時間をさらに確認できます。特定の環境、プロキシ、およびダッシュボードの期間を選択して、ダッシュボードをフィルタリングすることもできることも注目してください。



## 2. エラーコード分析

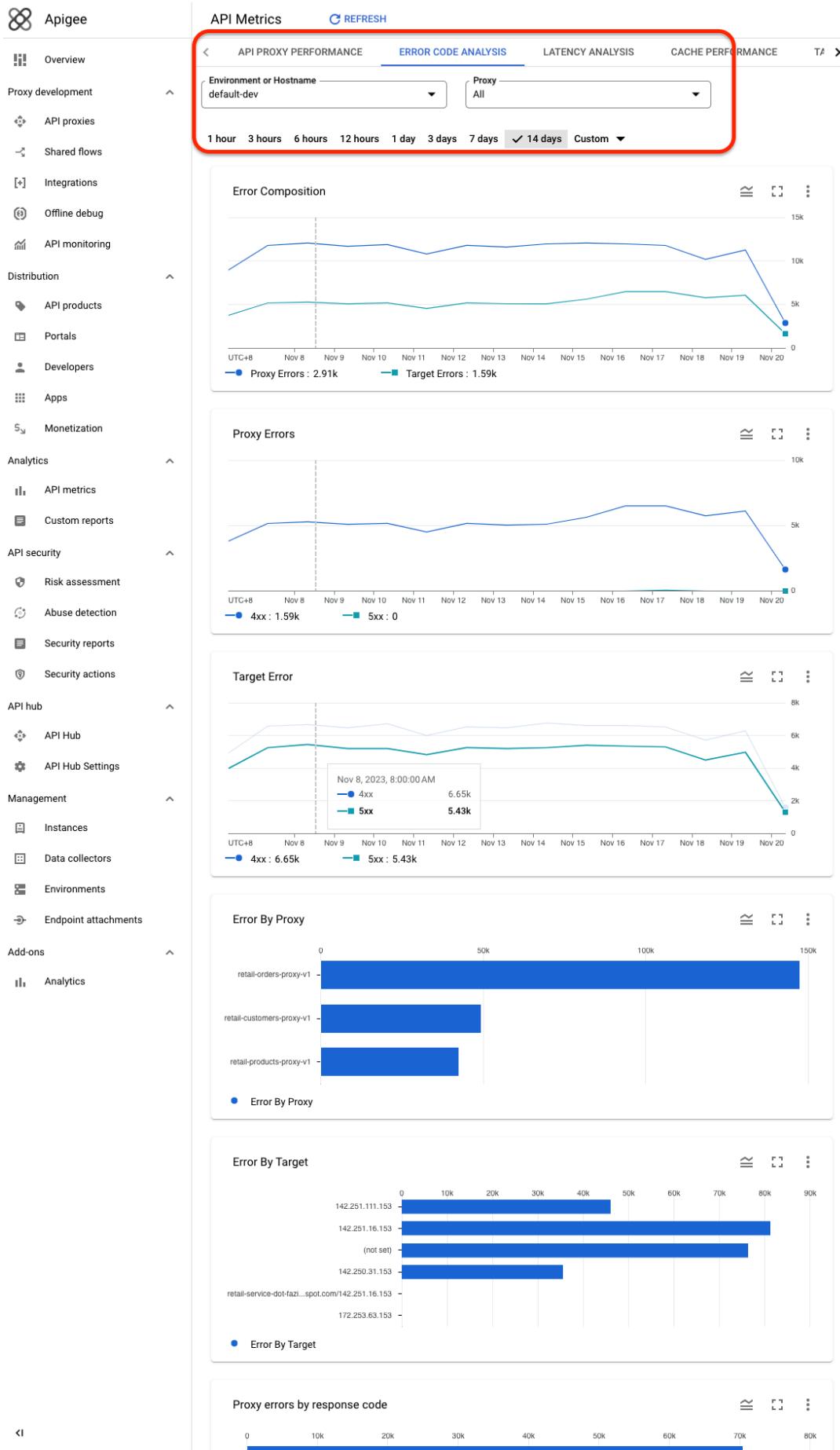
エラー コード分析ダッシュボードには、API プロキシとターゲットのエラー率が表示されます。エラー コード分析ダッシュボードでは次のものが使用されます:

- プロキシエラーを計算するためのレスポンスコード

- ターゲットエラーを計算するためのターゲットレスポンスコード

このダッシュボードでは、エラー構成、プロキシ エラー、ターゲットエラー、プロキシ別のエラー、ターゲット別のエラー、応答コード別のプロキシ エラー、応答コード別のターゲット エラーなどのダッシュボードを表示できます。

左側の同じ API metrics メニュー内で、[ERROR CODE ANALYSIS] タブを選択します。

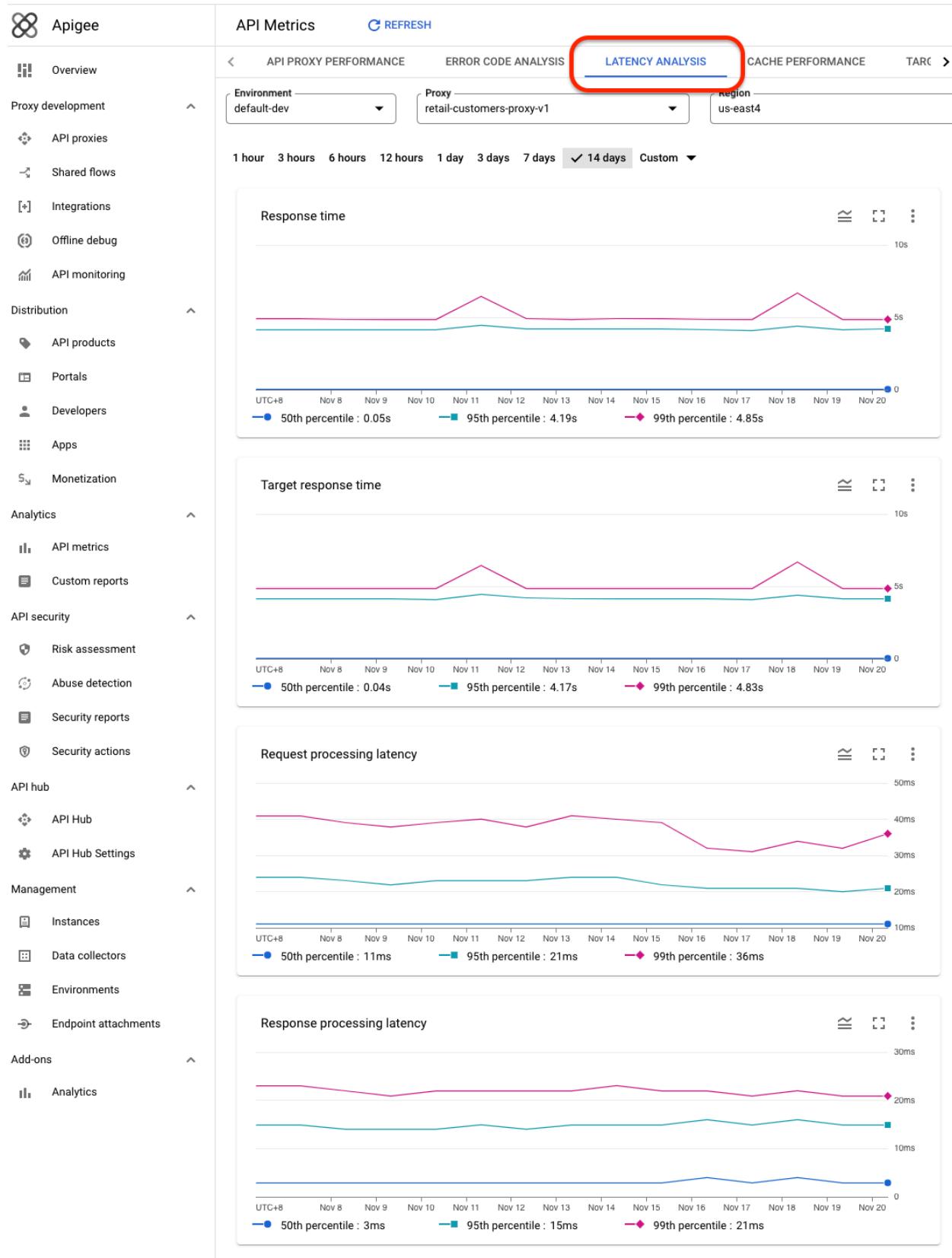


### 3. レイテンシー分析

レイテンシー分析ダッシュボードでは、API プロキシで発生している可能性のあるレイテンシーの問題を知ることができます。レイテンシーの測定値が 1 分単位まで表示され、中央値、95 パーセンタイル、および 99 パーセンタイルの値が強調表示されます。

これは、応答時間、目標応答時間、リクエスト処理レイテンシ、およびレスポンス処理レイテンシの 4 つのダッシュボードで構成されます。

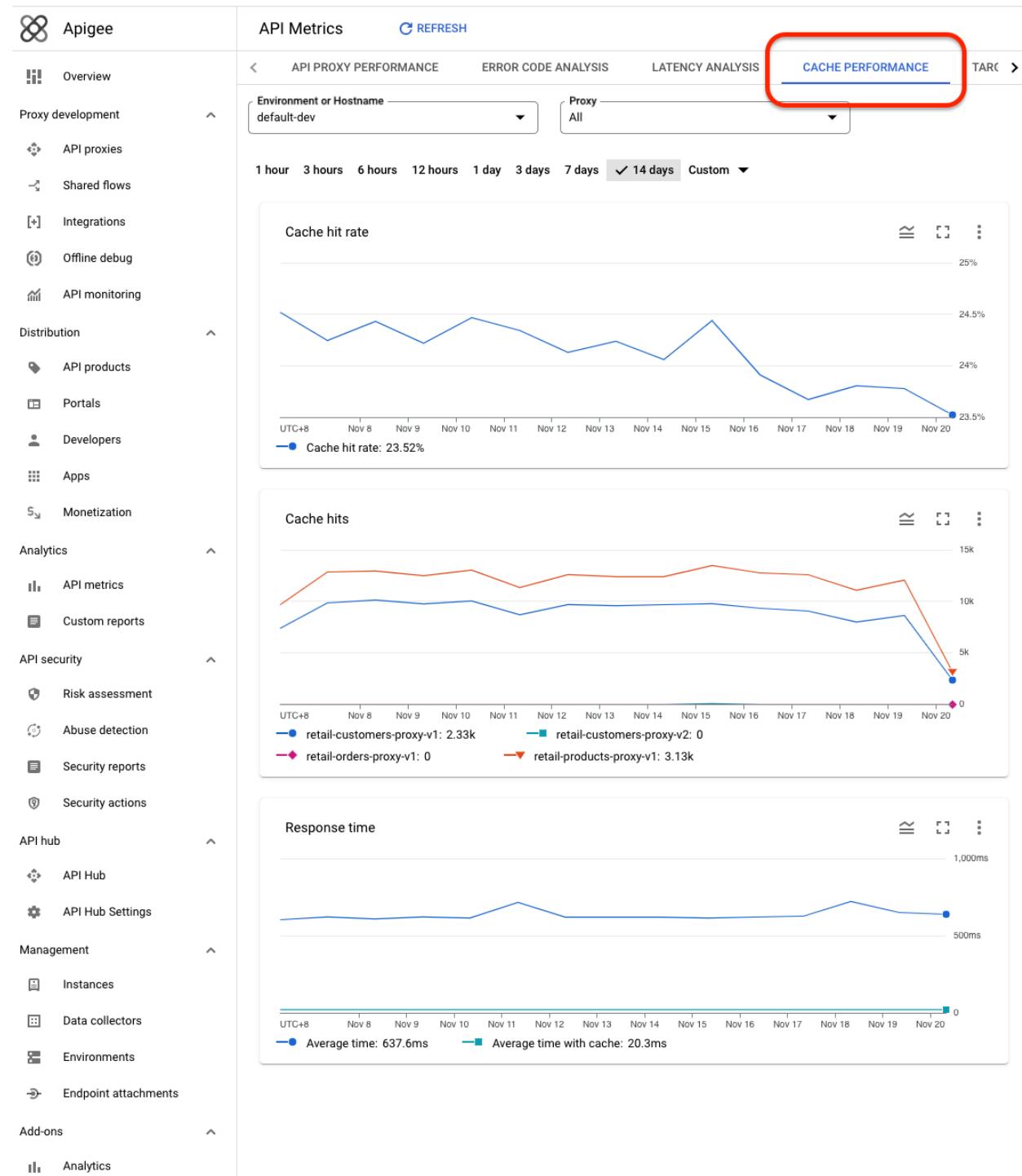
レイテンシー分析ダッシュボードは、[Error Code Analysis] タブのすぐ隣にあります。



#### 4. キャッシュパフォーマンス

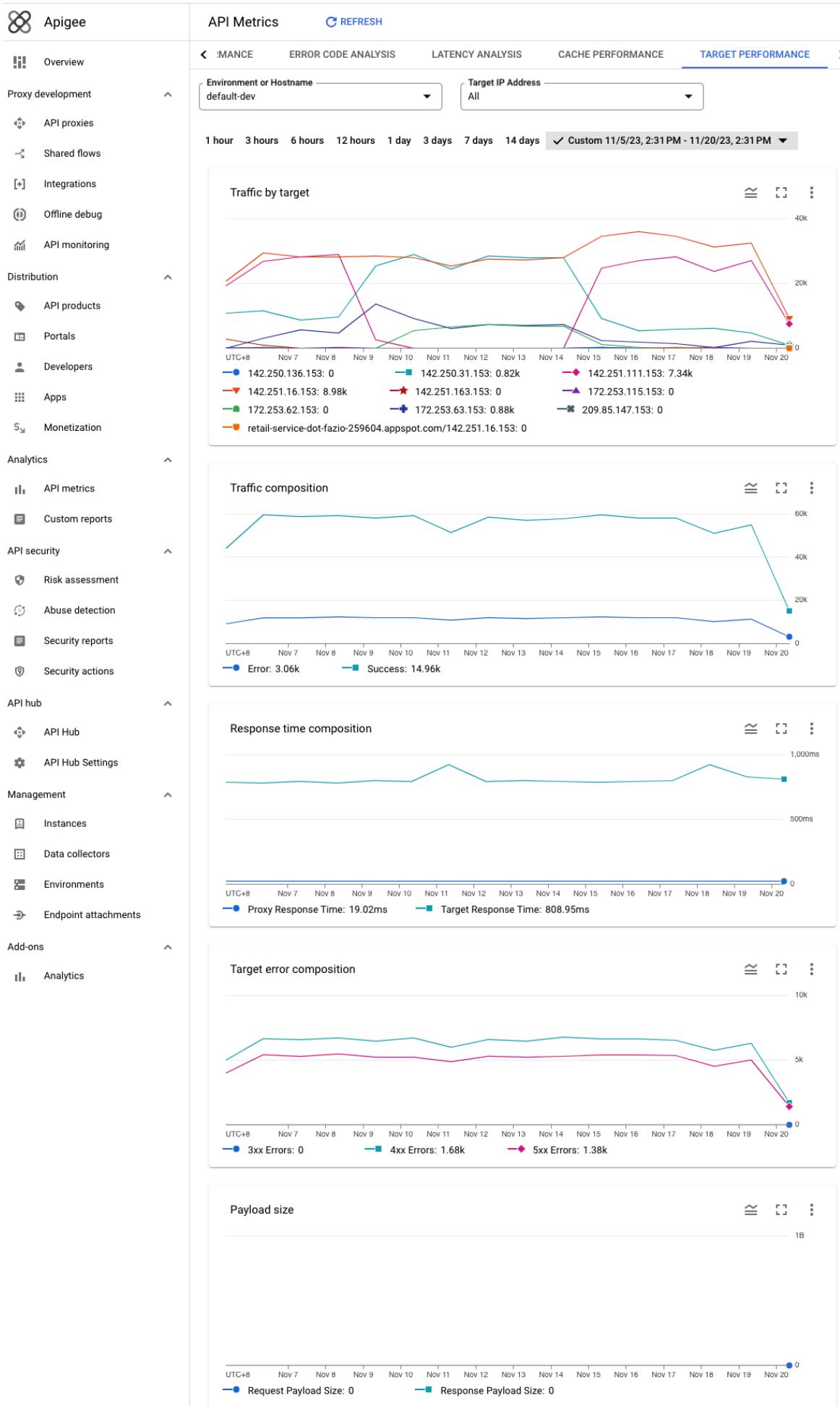
キャッシングパフォーマンスダッシュボードを使用すると、Apigee キャッシュの値を一目で確認できます。ダッシュボードは、バックエンド サーバーの待機時間の短縮と負荷の軽減という観点からキャッシングの利点を視覚化するのに役立ちます。

このダッシュボードは、キャッシングヒット率、キャッシングヒット、応答時間を測定します。



## 5. ターゲットパフォーマンス

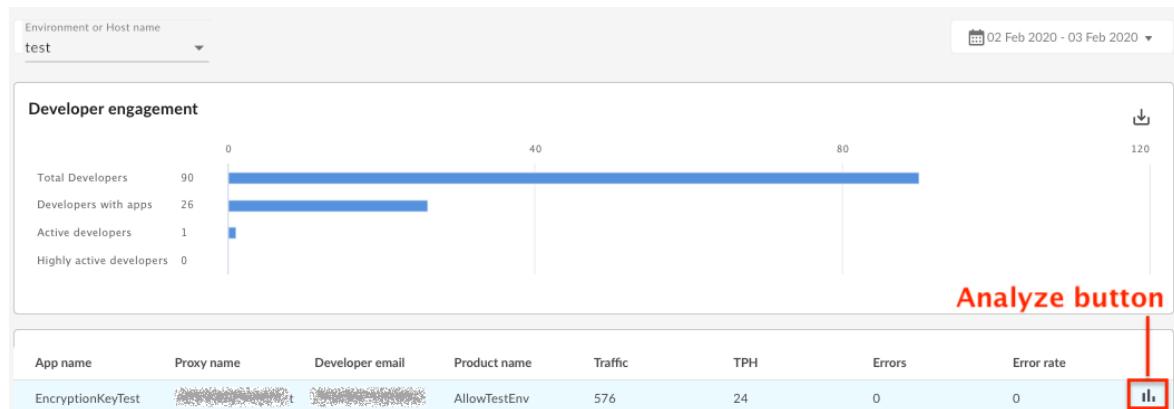
ターゲット パフォーマンス ダッシュボードは、API プロキシバックエンドターゲットのトラフィック パターンとパフォーマンス メトリックを視覚化するのに役立ちます。



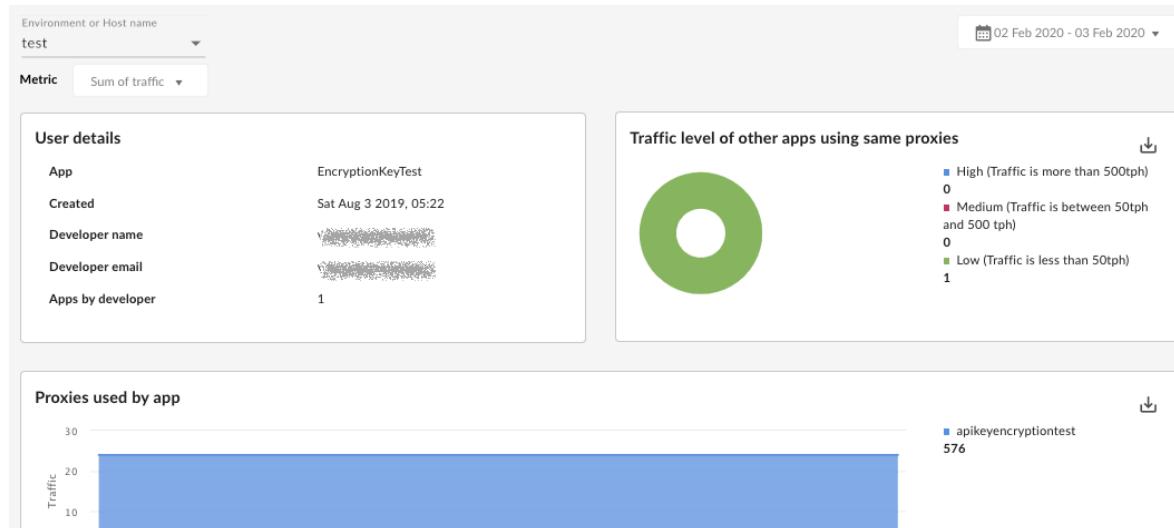
## 6. 開発者エンゲージメント

開発者エンゲージメントダッシュボードには、登録されているアプリ開発者ごとにどれだけ多くのAPIトラフィックを生成しているかがわかります。最も多くのAPIトラフィックと最も多くのエラーを生成している開発者を見つけることができます。たとえば、特定の開発者のアプリが他の開発者に比べて多くのエラーを生成している場合、その開発者と積極的に問題に対処できます。

注: 開発者エンゲージメントダッシュボードは、クラシック Apigee UI (<https://apigee.google.com>) の [Analyze] > [Developers] > [Developer Engagement] でのみ使用できます。



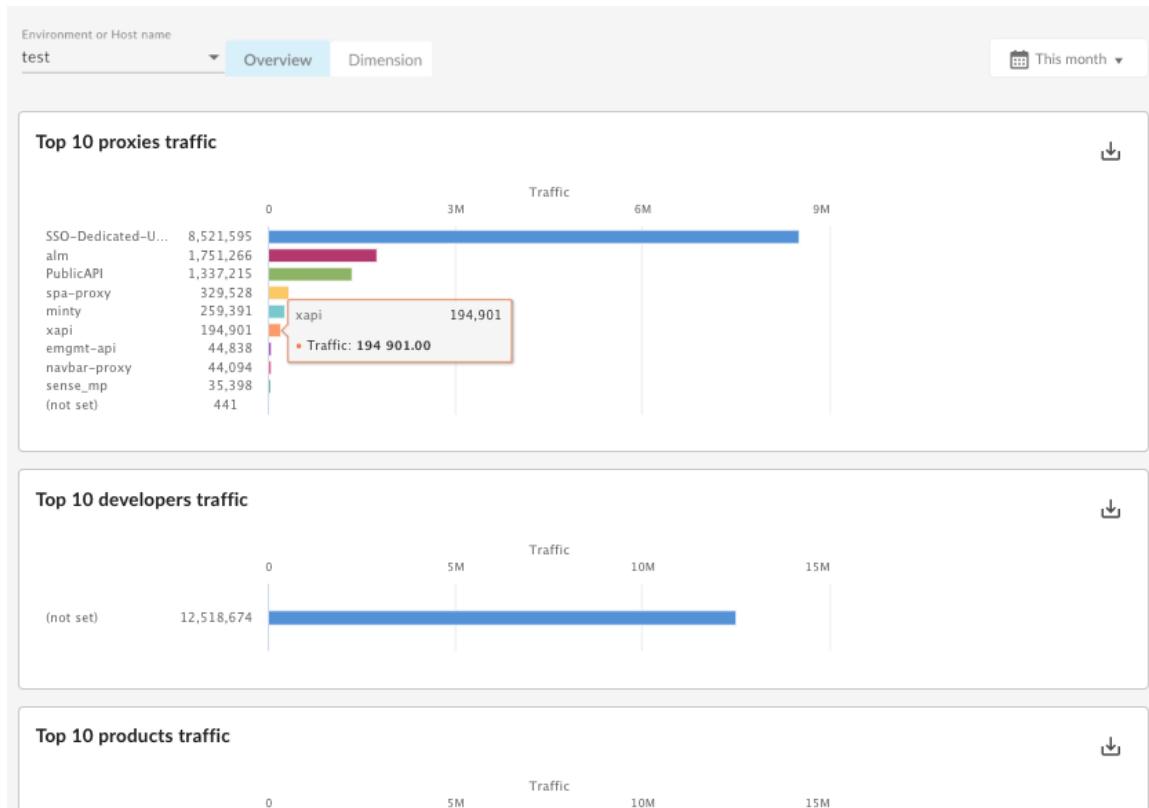
メインビューで、アプリの [Actions] 列の下にある [Analyze] ボタンを選択すると、そのアプリとアプリ開発者に関する詳細が表示されます(有効になっている場合):



## 7. トラフィック構成

トラフィック構成ダッシュボードは、API プログラム全体に対する上位の API、アプリ、開発者、製品の相対的な寄与を測定します。

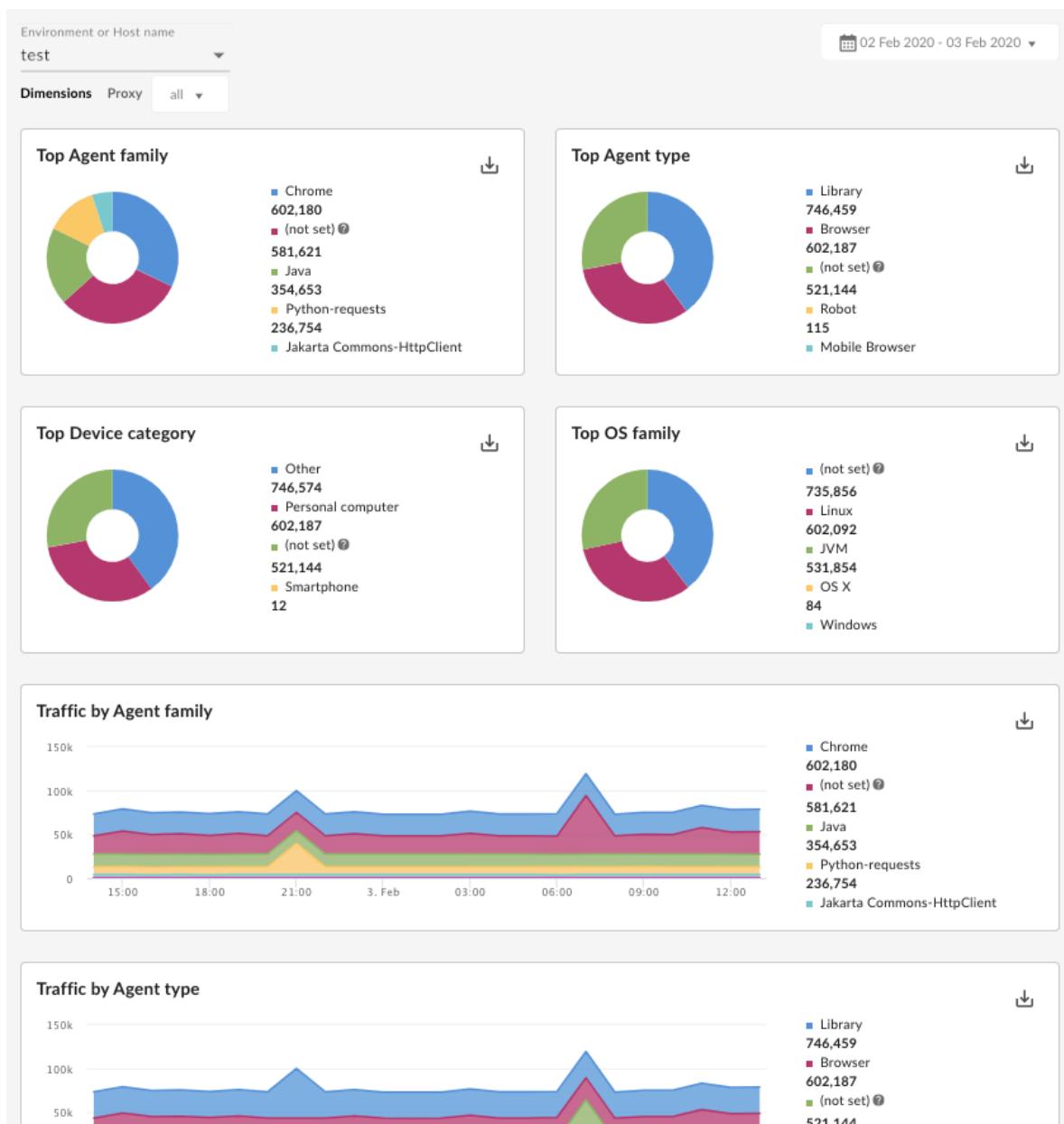
注: トラフィック構成ダッシュボードは、従来の Apigee UI (<https://apigee.google.com>) の [Analyze] > [Developers] > [Traffic Composition] でのみ使用できます。



## 8. デバイス

デバイス ダッシュボードには、API へのアクセスに使用されているデバイスとサーバーが表示されます。これにより、ユーザーが API にアクセスする方法の傾向を特定できます。たとえば、ある種類のデバイスからのトラフィックが増加している一方で、別の種類のデバイスからのトラフィックが減少していることに気づき、その変化に何らかのアクションが必要かどうかを判断する場合があります。

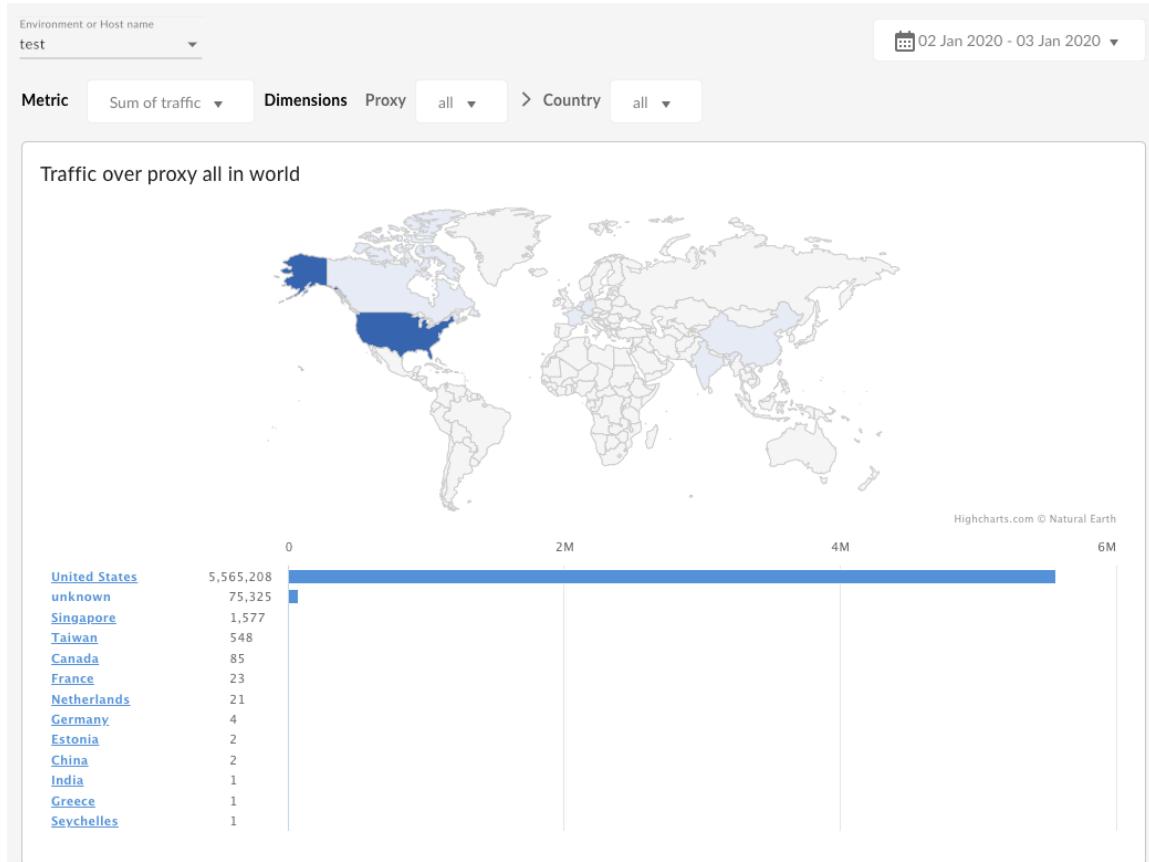
注: デバイス ダッシュボードは、クラシック Apigee UI (<https://apigee.google.com>) の [Analyze] > [End Users] > [Devices] でのみ使用できます。



## 9. ジオマップ

ジオマップ ダッシュボードは、地理的位置にわたるトラフィック パターン、エラー パターン、およびサービスの品質を追跡します。すべての API に関する情報を表示したり、特定の API にズームインしたりできます。

注: Geomap ダッシュボードは、従来の Apigee UI (<https://apigee.google.com>) の [Analyze] > [Developers] > [Geomap] でのみ使用できます。



## パート 5B - カスタム レポートの作成

すぐに使用できるレポートが要件を満たさない場合は、カスタム レポートを作成できます。

1. GCP Console の Apigee メニューに戻ります。Analytics → Custom Reports で、[+ CREATE] ボタンをクリックし、Custom Report を選択します。

The screenshot shows the 'Custom Reports' section of the Apigee Analytics menu. On the left, there's a sidebar with 'Custom reports' highlighted. In the main area, there's a table listing existing custom reports. The '+ CREATE' button is highlighted with a red box.

Display Name	Dimensions	Metrics	Last Modified	Actions
app traffic	API Product	sum(traffic)	Jul 29, 2023, 9:33:33 AM	⋮
Traffic by Station Id	Proxy dc_stationId	sum(traffic)	May 29, 2023, 2:54:36 PM	⋮
Response Time by Station Id	dc_stationId	avg(total response time) min(total response time) max(total response time)	May 29, 2023, 2:42:17 PM	⋮
Traffic by StationId	dc_stationId	sum(traffic)	May 29, 2023, 1:08:32 PM	⋮
app traffic	Developer App	sum(traffic)	Feb 9, 2023, 9:31:25 AM	⋮

2. 次のカスタム レポートの詳細を入力します。:

- Report Name: API Traffic by API Product and Client Location
- Report Description: API Traffic by API Product and Client Location
- **Column**のChart typeを選択します。

3. Metrics セクションの下:

- まず、**Aggregated function**にsum を指定し traffic を選択し [Done] をクリックします。
- [ADD A METRIC] をクリックし、**Aggregated function**にsum を指定し proxy errorsを選択して [Done] をクリックします。

4. Dimensions セクションの下:

- まず、[API Product] を選択し、[ADD A DIMENSION] をクリックします。
- 次に、[Country] を選択し、[ADD A DIMENSION] をクリックします。
- 最後にCityを選択します

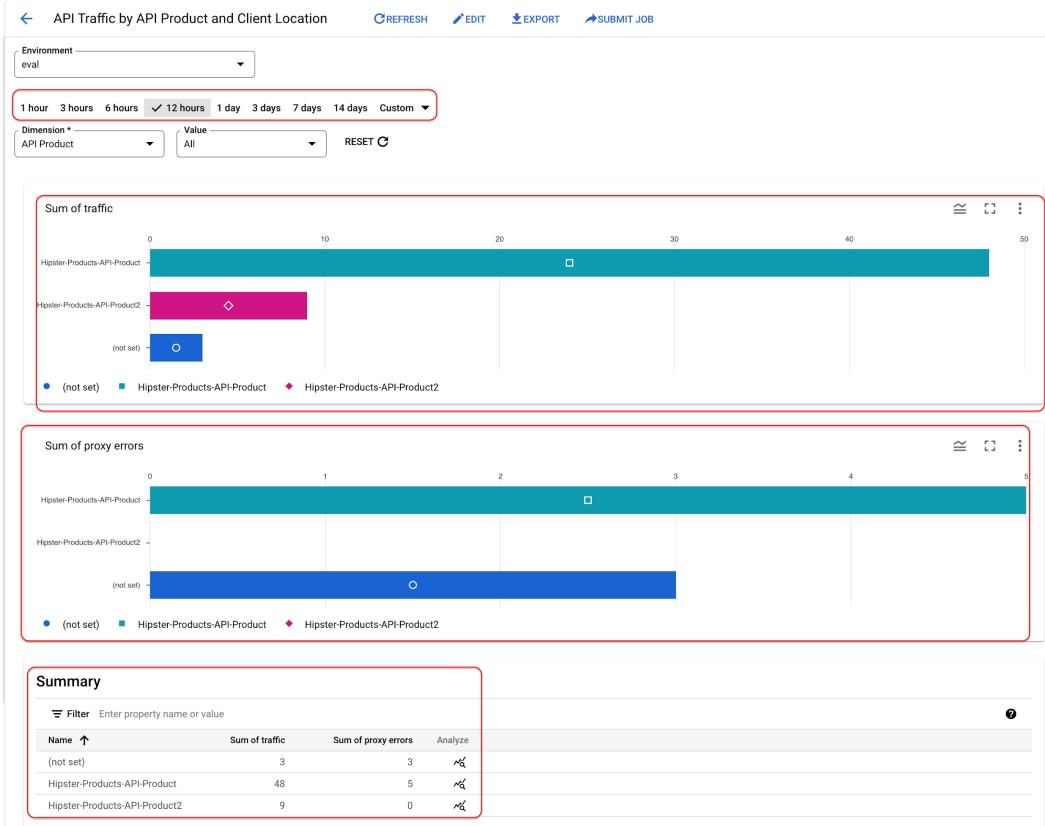
5. カスタム レポートは次のようにになります。

The screenshot shows the 'Create a custom report' page in the Apigee interface. The left sidebar lists various sections like Overview, Proxy development, Distribution, Analytics, and Management. The 'Custom reports' section is selected. The main area is titled 'Basics' and contains fields for 'Report name' (API Traffic by API Product and Client Location) and 'Report Description' (API Traffic by API Product and Client Location), both highlighted with a red box. Below this is the 'Chart Type' section, where 'Column' is selected (also highlighted with a red box). The 'Metrics' section is expanded, showing two metrics: 'Edit metric' (Select a metric: traffic, Aggregation function: sum) and 'New metric' (Select a metric: proxy errors, Aggregation function: sum). Both of these sections are also highlighted with a red box. The 'Dimensions' section shows three dimensions: Dimension 1 (API Product), Dimension 2 (Country), and Dimension 3 (City), all highlighted with a red box. At the bottom, there are 'CREATE' and 'CANCEL' buttons, with 'CREATE' being highlighted with a red box.

Click CREATE.

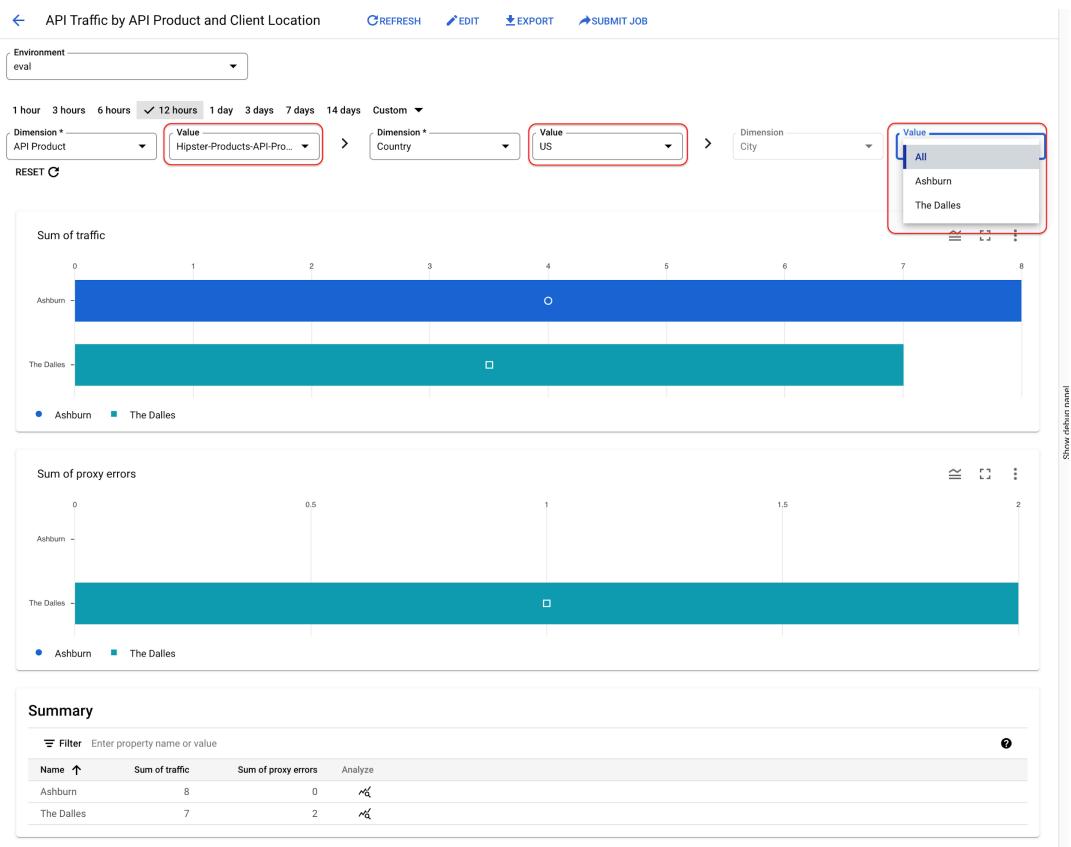
6. 作成したら、[API Traffic by API Product and Client Location] レポートをクリックしてレポートを表示します。カスタム レポートには、[Sum of traffic(トラフィックの合

計)] と [Sum of proxy errors(プロキシ エラーの合計)]を示す 2 つの縦棒グラフ図が表示されます。最後にサマリーセクションもあります。1 時間、3 時間などを選択してレポート時間を変更できることに注目してください。



7. 【オプション】さまざまな場所から API への複数のリクエストを生成します。簡単なヒントの 1 つは、GCP Cloud Shell または Web バージョンの Postman で CURL コマンドを使用することです。ただし、時間は刻一刻と過ぎているので、時間をかけすぎないようにしてください :)。
8. 作成中に 3 つの次元を選択したことに注意してください。ここで、API プロダクトの Dimension value セクションで API プロダクトのいずれかを選択すると、API クライアントが API にアクセスするCountryディメンションが表示されるはずです。いずれ

かのCountryのいずれかを選択すると、そのcityが表示されます。



9. 右上の [Export] ボタンをクリックしてデータを CSV としてダウンロードしたり、レポートの時間範囲を変更したりすることもできます。

## パート 5 - まとめ

このラボでは、API プログラムの成功とパフォーマンスを示し、API を最適化する方法についての貴重な洞察を提供するすべてのレポートを調べました。