Stripe & Next.js + AWS Amplify で会員 + 定期課金機能

JP_Stripes Online

Hidetaka Okamoto

#JP_Stripes

Hidetaka Okamoto

- **Digitalcube Co. Ltd.**
- **JavaScript Developer**
- **WordPress 4.7 / 5.0 / 5.3 Core contributor**

aws CERTIFIED

- Developer - Associate
- Solutions Architect - Associate
- SysOps Administrator - Associate
- DevOps Engineer - Professional

【本編無料】Stripe CLIの本

hidetaka okamoto

Stripeから提供されているCLIツールのインストール方法や使い方について紹介します。
JP_Stripesなどのコミュニティイベントで紹介する内容を本としてまとめたものですので、本編については無料公開としています。

有料部分には個人的に使っているTipsのようなものを載せていますので、興味がある方はあわせてご覧ください。

紹介しているCLI version
v1.5.8

前提としている環境
macOS / zsh

https://zenn.dev/hideokamoto/books/e961b4bad92429

Hidetaka Okamoto

- Digitalcube Co. Ltd.
- JavaScript Developer
- WordPress 4.7 / 5.0 / 5.3 Core contributor

aws CERTIFIED

- Developer - Associate
- Solutions Architect - Associate
- SysOps Administrator - Associate
- DevOps Engineer - Professional

Agenda

- 最小限の開発工数で有料会員メディアを作る
- Next.js / Stripe / AWS Amplifyで実装する
- 設計・開発時の注意点
- 「正しくSaaSに依存する」

#JP_Stripes

Agenda

- **最小限の開発工数で有料会員メディアを作る**
- Next.js / Stripe / AWS Amplifyで実装する
- 設計・開発時の注意点
- 「正しくSaaSに依存する」

#JP_Stripes

気になるツールや
新機能で遊ぶ場所が欲しい

個人サイトに全部ブチ込んだ

https://wp-kyoto.net で

9月中公開予定

#JP_Stripes

技術スタック

- AWS Amplify: Authentication
- AWS CDK: IaC
- Stripe: Subscription
- Algolia: Advanced Search
- Next.js: Framework
- Ionic: UI library
- WordPress: Headless CMS
- Capacitor / Sentry / etc…

#JP_Stripes

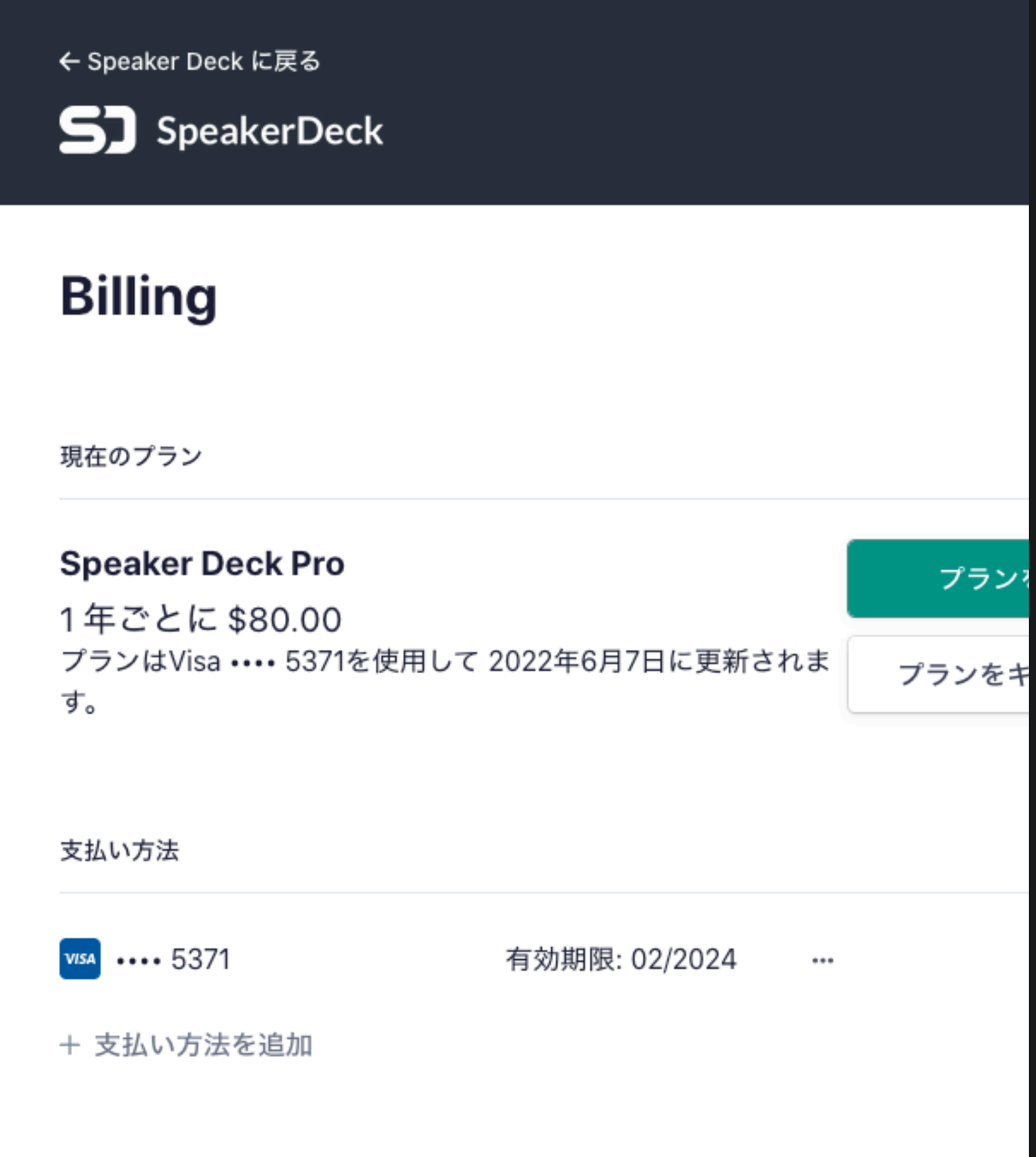
有料会員サイト開発のショートカットコース

- 複雑な機能はすべてSaaS / Frameworkに任せる
- アプリは「APIをよびだす」「APIの結果を表示する」に特化
- 「決済」と「認証認可」をどれだけ最短距離で作れるかが鍵
  - 認証認可: Auth0 / Cognito / Firebase / Superbase / etc…
  - 決済: Stripe / pay.jp / PayPal / Paidy / etc…

#JP_Stripes

Stripeの Low code系機能を活用

#JP_Stripes



SpeakerDeck

#JP_Stripes

SaaS APIを活用して、コードの量を抑える

- 決済と決済管理はほぼ必ず必要になる機能要件
- Stripeで決済する場合、以下の2点がLow Code化できる
  - 注文フォーム(カード / 購入者情報入力) -> Checkout
  - 定期課金情報管理 -> Customer Portal
- 「自由と効率」どっちを重視するかで道を選ぼう

Agenda

- 最小限の開発工数で有料会員メディアを作る
- **Next.js / Stripe / AWS Amplifyで実装する**
- 設計・開発時の注意点
- 「正しくSaaSに依存する」

#JP_Stripes

Next.js & AWS AmplifyにStripeを組み込む

- React App側にStripe JS SDK
- REST API側にStripe Nodejs SDK
- CheckoutとCustomer Portalだけなら React SDKは不要
  - React SDKはStripe Elementを使うためのSDK
- APIキーの取り扱いに注意（後述）

#JP_Stripes

SSMまたはSecret ManagerからAPIキーを取得

```
export const createStripeClient = async () => {
  const data = await new SSM({ region: “us-east-1" }) 
    .getParameter({
      Name: !!process.env.AWS_LAMBDA_FUNCTION_NAME ? "STRIPE_LIVE_SECRET_KEY": 
"STRIPE_TEST_SECRET_KEY",
      WithDecryption: true,
    }).promise();

  const stripe = new Stripe(data.Parameter?.Value;, {
    apiVersion: "2020-08-27",
    maxNetworkRetries: 3,
  });
  return stripe;
};
```

#JP_Stripes

Stripe CustomerとCognito User poolの連携

```
// Authorization Headerなどで取得したtokenでCognito UserをGet
const user = await cognito.getUser({
        AccessToken: req.headers.authorization || “"
}).promise()


// User attributesからEmailを取得
const emailAttribute = user.UserAttributes.find((data) => data.Name === "email");
const email = emailAttribute?.Value || “";

// Emailをつかってstripe Customerを作成
const customer = await stripe.customers.create({ email });
```

#JP_Stripes

Stripe CustomerとCognito User poolの連携

```
// 作成したStripe Customer
const customer = await stripe.customers.create({ email });

// User Attributeに保存する
await cognito
  .updateUserAttributes({
    AccessToken: req.headers.authorization,
    UserAttributes: [{
        Name: "custom:stripeCustomerId",
        Value: customer.id,
      }],
  }).promise();
```

Customer idをセットしてcheckout.sessions.create

```
const session = await stripe.checkout.sessions.create({
  customer: customerId,
  allow_promotion_codes: true,
  line_items: [{
      price: price.id,
      quantity: 1,
  }],
  mode: "subscription",
  success_url: `${appUrl.replace(/\/$/, "")}/mypages/subscriptions`,
  cancel_url: `${appUrl.replace(/\/$/, "")}/mypages/plans`,
  payment_method_types: ["card"],
});

res.status(200).json({ session_id: session.id });
```

#JP_Stripes

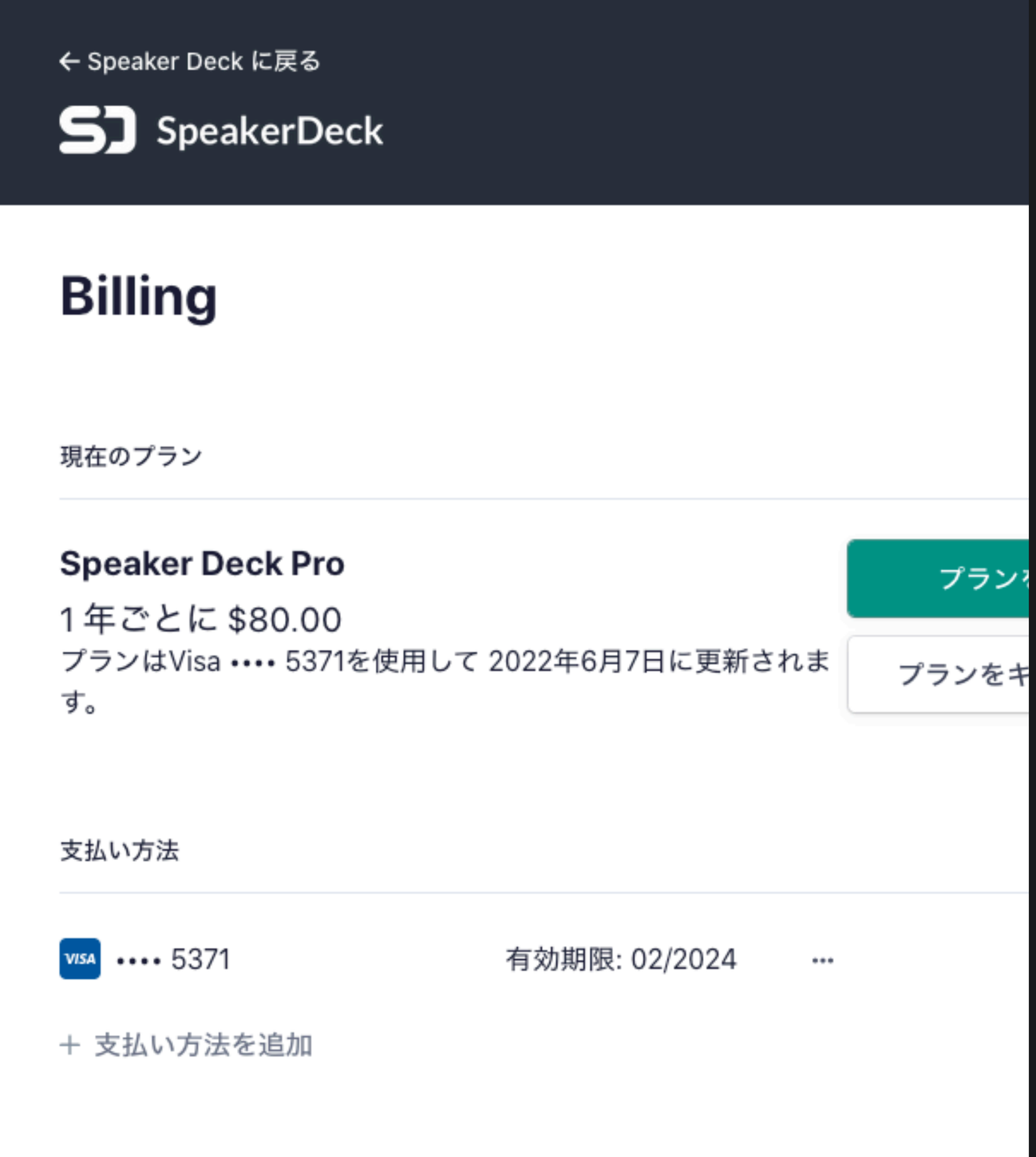
Customer PortalはCustomer IDを使う

```
// Cognitoからユーザーを取得
const user = await cognito.getUser({ AccessToken: req.headers.authorization }).promise()
const attribute = user.UserAttributes.find((data) => {
  return data.Name === “custom:stripeCustomerId"
});

// Customer Poral Sessionを作成
const session = await stripe.billingPortal.sessions.create({
  customer: attribute?.Value,
  return_url: `${appUrl.replace(/\/$/, "")}/mypages/subscriptions`,
});

res.status(200).json({ url: session.url });
```

#JP_Stripes



面倒な
請求管理系画面が
たった4行で作れる

#JP_Stripes

APIでポータルのURLを生成してリダイレクト

```
const { push } = useRouter();
return (<button onClick={async () => {
          const session = await Auth.currentSession();
          const token = session.getAccessToken().getJwtToken();
          const data = await fetch("/api/portal", {
            method: "POST",
            headers: {
              "Content-Type": "application/json",
              Authorization: token,
            },
          }).then(data => data.json());
          push(data.url);
      }}>Button</button>)
```

#JP_Stripes

実装時のポイント

1. StripeのCustomer IDをユーザー情報と紐づける必要あり
   - Customer Portal Linkの生成はサーバー側処理
2. Customer Portalに出すコンテンツの制御は原則Dashboard
   - APIからなら設定は複数作れる模様（未検証）
3. Stripe Secret APIキーの扱いが不安なら制限付きキーを使おう

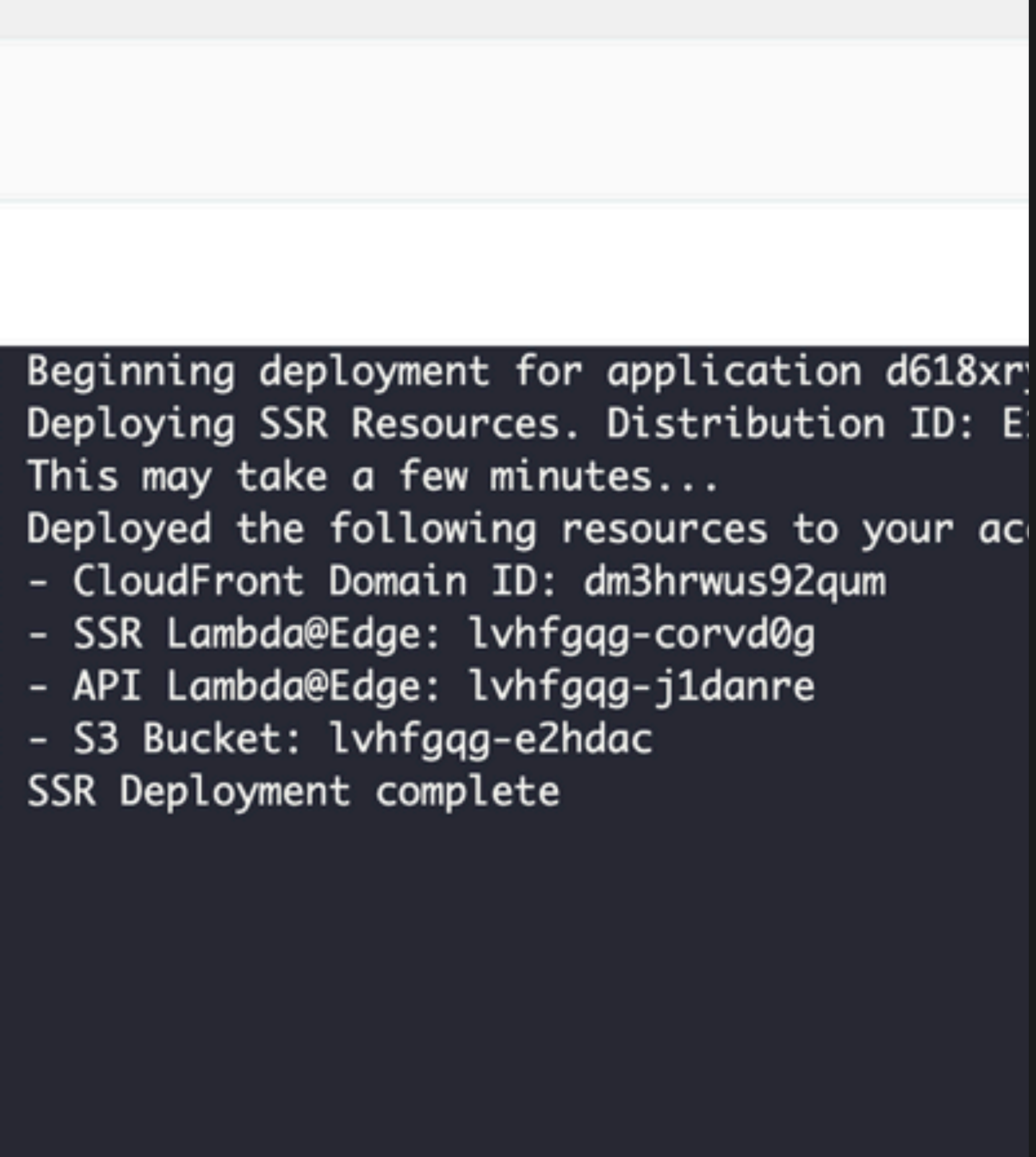
Agenda

- 最小限の開発工数で有料会員メディアを作る
- Next.js / Stripe / AWS Amplifyで実装する
- **設計・開発時の注意点**
- 「正しくSaaSに依存する」

#JP_Stripes

利用規約ページと
プライバシーポリシーページ
は登録必須

#JP_Stripes



AWS Amplifyは
Next.jsをLambda@edge
で実行する

#JP_Stripes

だいたい
これだと思って
扱おう

アプリ外でStripeのデータが変わることに注意

- プラン変更・解約・ユーザー情報の変更などの操作
- Customer Portalで起きた変更をシステムに反映させる必要がある
  - Stripe Webhookでイベントを受けてデータを変更しよう
- Customer Portalには「Customer削除」が現状ない
  1. Subscriptions.delete Webhookでデータを消す
  2. Cognitoなどのユーザー削除時に消すようにシステムを構築

#JP_Stripes

必要なAPIを呼び出して
機能を開発する

↓

必要なWebhookイベントを
Subscribeして開発する

Agenda

- 最小限の開発工数で有料会員メディアを作る
- Next.js / Stripe / AWS Amplifyで実装する
- 設計・開発時の注意点
- **「正しくSaaSに依存する」**

#JP_Stripes

設定項目は日に日に増加している

支払い方法

顧客による支払い方法の更新を許可

プロモーションコード

顧客によるプロモーションコードの適用を許可します。もっと知る

サブスクリプションをキャンセル

顧客によるサブスクリプションのキャンセルを許可

- 今すぐキャンセル
  サブスクリプションを今すぐキャンセルします。
  - サブスクリプションのキャンセルによる日割り計算
- 請求期間の終了時にキャンセル
  キャンセル後も、顧客は請求期間の終了まではサブスクリプションを更新できます。

キャンセル理由を収集する

理由を編集

サブスクリプションの一時停止

顧客によるサブスクリプションの一時停止と再開を許可

サブスクリプションを更新

顧客による別の料金プランへの変更を許可

顧客によるサブスクリプションの数量の更新を許可

Customer Portalと
同等の機能を
自前で開発・提供できるか？

コアビジネス領域外の機能を誰が作るか？

A. 自力開発することで、デザインや仕様の自由度を確保できる

- 機能が増えると保守・運用の範囲も増える
- その自由を活用できるだけの開発リソースがあるか否か

B. SaaS / Low Codeに委譲して、重要な機能に集中する

- ベンダーが最適とするUIや機能を常に受け取れる
- ドメインやデザイン・機能の制約を許容できるか否か

イベント駆動なシステムだからできること

- Checkout / Customer PortalはWebhookでシステムと連携する仕組み
  - プラン変更などのイベントを受けて動作するシステムになる
- 「**同じイベントを発火できれば**」いつでも置き換えできる
- 立ち上げ時は両機能を使ってコア開発にリソースを集中する
- 移行する時は、**イベント単位で段階的に**切り替えていく
- 「イベント駆動アーキテクチャ」への第一歩に

#JP_Stripes

More info

oper Digest <dev-digest@e.stripe.com> Sent on: Thu, 04/22/

developer foundations videos for our client libraries

o

stripe

April Developer Digest

Get started quickly with new foundations videos for our client libraries

Our new video series covers the fundamentals of topics like authentication, making requests, metadata, pagination, webhook helpers, and more so you can get started quickly with any of our official client libraries.

Check out the playlist for your preferred language:

メールで更新情報をざっと見できる

Developer Digest

https://stripe.dev/

#JP_Stripes

Stripe Developers

チャンネル登録者数 1.1万人

チャンネル登録

ホーム　動画　再生リスト　コミュニティ　チャンネル　概要

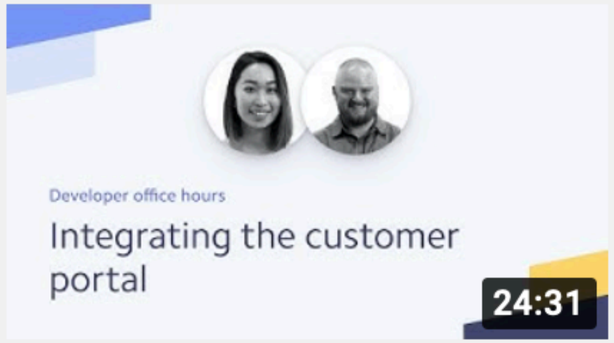
Developer office hours

▶ すべて再生

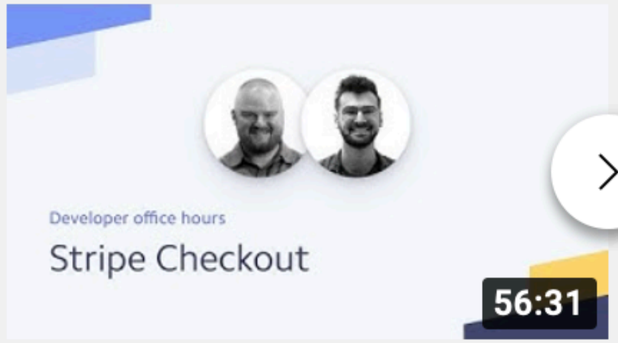
Stripe engineers answering questions and walking through code examples. Visit the GitHub repo to see the code: https://github.com/stripe-samples/developer-office-hours

React Stripe.js

Stripe Checkout



Integrating the customer portal



Stripe Checkout

https://www.youtube.com/stripedevelopers

Thanks!

- 最小限の開発工数で有料会員メディアを作る
- Next.js / Stripe / AWS Amplifyで実装する
- 設計・開発時の注意点
- 「正しくSaaSに依存する」

#JP_Stripes